

NPS ARCHIVE
1969
ARICK, J.

A COMPUTER PROGRAM FOR INTEGER
SOLUTIONS TO LINEAR PROGRAMMING
PROBLEMS

by

John Chaney Arick

**DUDLEY KNOX LIBRARY
NAVAL POSTGRADUATE SCHOOL
MONTEREY, CA 93943-5101**

United States Naval Postgraduate School



THESIS

A COMPUTER PROGRAM FOR
INTEGER SOLUTIONS TO LINEAR PROGRAMMING PROBLEMS

by

John Chaney Arick

October 1969

T 132 208

This document has been approved for public release and sale; its distribution is unlimited.

Library

U.S. Naval Postgraduate School

Monterey, California 93940

A Computer Program for
Integer Solutions to Linear Programming Problems

by

John Chaney Arick
Captain, United States Marine Corps
B.S., U. S. Naval Academy, 1962

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
October 1969

NPS ARCHIVE ~~Thesis~~ A66 c.1
1969
ARICK, J

ABSTRACT

An algorithm for the solution of integer linear programming problems is presented and programmed in Fortran IV for use on digital computers. The program incorporates an optional feature which provides all existing alternative optimal solutions. Solutions, computation times, and iteration requirements for each of thirteen test problems are summarized and discussed.

TABLE OF CONTENTS

I.	INTRODUCTION -----	9
II.	THE ALGORITHM -----	11
III.	PROGRAMMING CONSIDERATIONS -----	19
IV.	TEST PROBLEMS -----	25
V.	COMPUTATIONAL RESULTS -----	30
VI.	CONCLUSIONS -----	38
	APPENDIX A Input Data and Output Formats -----	40
	APPENDIX B Overlay Structure -----	42
	COMPUTER PROGRAM -----	44
	BIBLIOGRAPHY -----	72
	INITIAL DISTRIBUTION LIST -----	73
	FORM DD 1473 -----	75

LIST OF TABLES

I.	ADDITIONAL TEST PROBLEM RESULTS -----	31
II.	SOLUTIONS -----	34
III.	SECTIONS REQUIRED AND COMPUTATION TIMES -----	37
IV.	A COMPARISON OF ITERATION REQUIREMENTS -----	39

LIST OF FIGURES

FIGURE 1	THE OVERLAY TREE -----	43
----------	------------------------	----

I. INTRODUCTION

An algorithm presented by Dr. Harold Greenberg of the Naval Post-graduate School [1] provides an iterative procedure for the solution of the integer linear programming program: find

$$X = (x_1, x_2, \dots, x_n) \text{ which}$$

minimizes

$$C X \tag{1}$$

subject to

$$AX = B$$

where

$$0 \leq x_j \leq b_j, x_j \text{ integer, } j=1, \dots, n.$$

C is an n-vector, B is an m-vector, A is m x n, and the components of C, B, and A, as well as the b_j are integer.

A bounded variable linear programming algorithm [2] is first utilized to transform (1) into its equivalent problem (i.e., a continuous solution to (1)):

minimize

$$d + \sum_{j \in H} c_j x_j$$

subject to

$$x_G + \sum_{j \in H} \alpha_j x_j = \alpha_0 \tag{2}$$

where

$$0 \leq x_j \leq b_j, x_j \text{ integer, } j=1, \dots, n.^1$$

¹The first five subroutines of the program accomplish this portion of the solution and were provided by Dr. Greenberg.

G is the set of indices of basic variables; H is the set of indices of non-basic variables; x_G , α_j , and α_0 are vectors, with x_G the vector of non-basic variables.

If α_0 is not all integer, the equivalent Knapsack Problem [3] is then formed:

minimize

$$\sum_{j \in H} c_j x_j$$

subject to

(3)

$$\sum_{j \in H} \beta_j x_j = \beta_0 \pmod{1}$$

where $0 \leq x_j \leq b_j$, x_j integer.

The β 's are the fractional parts of the α 's from (2) above, and as such are also vectors.

The Knapsack problem is subsequently solved [1] using a dynamic programming enumeration which guarantees that all existing integer solutions can be found. Should there exist no feasible integer solution, the iterative process ceases after all possible column vectors of fractional parts have failed to produce a feasible solution.

II. THE ALGORITHM²

1. Let the indices, H , be $1, 2, \dots, m$. Form the following tableau (subsequently referred to as a section):

$1, 2, \dots, m$
$C_1 \ C_2 \qquad \qquad \qquad C_m$
$\beta_1 \ \beta_2 \qquad \qquad \qquad \beta_m$
$x_j=0$

and go to 2.

2. (a) Given the (new) section, test for $\beta_j = \beta_0$ for all $j \in H$. If, for one or more j , $\beta_j = \beta_0$, select the minimum C_j and call this C_r . If $C_r < C_r^*$, where C_r^* was the C_j which produced the last feasible solution, set $C^* = C_r$ and go to 4.

(b) If, for all unmarked j in the current section, $\beta_j \neq \beta_0$, find $C_r = \min C_j$ for all unmarked columns in all sections. If $C_r = C_r^*$, the current solution corresponding to C_r^* is the optimal solution to (1). If $C_r < C_r^*$, mark the column and go to 3.

3. (a) Form a new section in the following manner: compute $C_j' = C_r + C_j$, where the C_j are the values from the first section. If $C_j' \geq C_j^*$ do not add a corresponding column to the new section. Otherwise, continue by computing $\beta_j' = (\beta_r + \beta_j) \bmod 1$ for all $j \in H$ (the β_j are the values taken from section 1). For the $x_j = b_j$, $j \neq r$, and $x_r + 1 = b_r$ for $j = r$, for the section containing the newly marked column, do not add

²The majority of procedures presented here are taken from the original algorithm [1].

the corresponding column to the new section. For columns which duplicate those already formed and for which the subsequent use of either column would result in identical values for the x_j , do not add a column.

(b) Underneath the new section, write the x_j values from the section containing the newly marked column. Increase x_r by one for the new section and go to 2.

4. Take as a trial solution the values of the variables found below the section where $\beta_r = \beta_0$ with x_r increased by one. If the constraints in (2) are satisfied, replace the current feasible solution corresponding to the old C_r^* with the new solution and go to 2 (b). If they are not, replace the C_r^* with the previous value and go to 2 (b).

Example:³

minimize

$$5x_1 + 7x_2 + 10x_3 + 3x_4 + x_5$$

subject to

$$\begin{aligned} x_1 - 3x_2 + 5x_3 + x_4 - 4x_5 &\geq 2 \\ -2x_1 + 6x_2 - 3x_3 - 2x_4 + 2x_5 &\geq 0 \\ -x_2 + 2x_3 - x_4 - x_5 &\geq 1 \end{aligned}$$

$$0 \leq x_j \leq 1, x_j \text{ integer}, j=1, \dots, 5.$$

The surplus variables are added ($x_j \geq 0, j=6,7,8$) and the problem is solved for its continuous solution which yields the equivalent problem:

minimize

$$9 + 93/9 x_1 + 156/9 x_4 + 42/9 x_5 + 24/9 x_7 + 81/9 x_8$$

³This problem originally appeared as an example problem in [1] and was also used as test problem #10 for the program presented here.

subject to

$$-7/9 x_1 - 28/9 x_4 + 13/9 x_5 + x_6 + 1/9 x_7 - 21/9 x_8 = 3/9$$

$$-2/9 x_1 + x_3 - 8/9 x_4 - 4/9 x_5 - 1/9 x_7 - 6/9 x_8 = 6/9$$

$$-4/9 x_1 + x_2 - 7/9 x_4 + 1/9 x_5 - 2/9 x_7 - 3/9 x_8 = 3/9$$

$$0 \leq x_j \leq 1 ; j=1, \dots, 5; x_j \leq 0, j=6,7,8; x_j \text{ integer.}$$

Section (1) is formed as follows:

1	4	5	7	8
93	156	42	24	81
2	8	4	1	6
7	1	5	8	3
5	2	1	7	6
*	*	*	*	*

$x_j = 0$

(1)

with $\beta_0 = (3, 6, 3)$

Since $\min C_j = 24$, $C_r = 24$ and $r = 7$. Mark column 7 in section (1) and form the new section:

1	4	5	7	8
117	180	66	48	105
3	0	5	2	7
6	0	4	7	2
3	0	8	5	4
*	*	*	*	*

$x_7 = 1$

(2)

Note that in column 1 of section (2) $\beta_1 = \beta_0$. This means that an integer solution is obtained by substituting the non-basic variable values found below section 2 (with x_r , i.e., x_1 , increased by one) into the equivalent problem objective function and constraints.

This yields the following solution:

$$z^* = 22, x_6 = 1, x_3 = 1, x_2 = 1, x_1 = 1, x_2 = 1,$$

which satisfies the feasibility requirements. Set $C^* = C_1$, i.e., $C^* = 117$ (C^* can initially be considered ∞). At this point, mark all columns for which $C_j \geq C^*$ (but not the C^* column itself) and select the next $\min C_j = 42$. Mark column 5 of section (1) and form the new section:

1	4	8
135	198	147
6	3	1
3	6	8
6	3	7
*	*	*

$x_5=1$

Note that columns 5 and 7 have not been formed. The former because x_5 is now at its upper bound and can no longer be increased, the latter because it would duplicate column 5 of section (2) (the use of either will result in identical non-basic variable values). Also since $C^* = 117$, this entire section can be eliminated; use of any of its columns will result in a higher objective function value.

Select the next $\min C_j = 48$, mark column 7 of section (2) and form the new section:

5	7
90	72
6	3
3	6
6	3
*	

$x_7=2$

(3)

Since $C_j > C^*$, $j = 1, 4, 8$, the corresponding columns have been omitted.
 Since column 7 has $\beta_7 = \beta_0$, for the integer solution:

$$z^* = 17, x_6 = 0, x_3 = 1, x_2 = 1, x_7 = 3.$$

which satisfies the feasibility requirements. $C^* = 72$ now replaces $C^* = 117$ and all previously formed columns for which $C_j > C^*$ are marked.

Select the next $\min C_j = 66$, mark column 5 of section (2), and form the new section:

1	4	8
159	222	147
7	4	2
2	5	7
4	1	2
*	*	*
$x_7=1$		$x_5=1$

But $C_j > C^*$ for all columns of this section, so delete the entire section (column 5 had been omitted since x_5 is at its upper bound; column 7 had been omitted since it was a duplicate meeting the requirements for the omission of duplicates mentioned above). Select $\min C_j = 72$ and note that $C_r = C^*$ which means the solution corresponding to $C^* = 72$ is optimal.

The algorithm as originally presented in [1] makes no provision for the recognition of an integer L.P. problem which, although it has an optimal feasible continuous solution, has no feasible integer solution. This means that such a problem, when operated on by a computer program utilizing the algorithm, will run indefinitely, or until space and/or time limitations are reached. This of course can be costly in terms of

computer time, so a method is presented here, and incorporated in the program, for recognizing such infeasibility conditions.⁴

Let

Δ = the common denominator of the coefficients
of the equivalent problem,

g = (a, b) be the greatest common denominator of
 a and b ,

$\beta_j^T = (a_{1j}, a_{2j}, \dots, a_{mj})$, $j = 1, \dots, n$, be the
 j^{th} vector of fractional parts of the first
section,

$g_{ij} \equiv (a_{ij}, \Delta)$,

$g_j \equiv \min_{1 \leq i \leq m} g_{ij} \quad j = 1, \dots, n$,

$p_j \equiv \Delta/g_j$ for all j , and

C_j = cost of $\vec{a}_j$.

Since the period, p_j , is actually the number of times column j can be added to any other column before the column j assumes its original values (mod Δ), it follows that

$$\vec{a}_j + p_j \vec{a}_j = \vec{a}_j \pmod{\Delta}, \text{ or}$$

$$p_j \vec{a}_j = \vec{0} \pmod{\Delta}.$$

Therefore, the maximum number of distinct column vectors, M , which can be formed by the algorithm will be

$$M = \prod_{j=1}^n p_j ;$$

this is because each column will assume p_j different intermediate values

when operated on by any other column, either consecutively or in combination with other columns, prior to reassuming its original values.

Note now that any vector, $\vec{K}$, generated by the algorithm can be written in the form

$$\vec{K} = \vec{a}_j + \sum_{\ell=1}^n \gamma_{\ell} \vec{a}_{\ell} \pmod{\Delta}$$

where $0 \leq \gamma_{\ell} \leq p_{\ell}$ for all ℓ . γ_{ℓ} is necessarily integer and determined by the number of times column ℓ was utilized in the formation of $\vec{K}$.

Note also that the individual $\gamma_{\ell} \vec{a}_{\ell}$ terms are actually the intermediate columns which were formed prior to $\vec{K}$ and which played a role in the formation of $\vec{K}$.

Since there is a definite upper bound on the number of distinct column vectors, a bound may now be placed on the costs which will be generated. The cost associated with any vector generated by the algorithm will be

$$C_j + \sum_{\ell=1}^n \gamma_{\ell} C_{\ell}.$$

The last distinct column vector to be formed prior to repetition will be

$$\sum_{j=1}^n (p_j - 1) \vec{a}_j,$$

and the cost associated with this is

$$\sum_{j=1}^n (p_j - 1) C_j.$$

By previous development, if we add any vector to the current one, say $j=\sigma$,

$$\sum_{j=1}^n (p_j - 1) \vec{a}_j + \vec{a}_{\sigma} = \sum_{\substack{j=1 \\ j \neq \sigma}}^n (p_j - 1) \vec{a}_j \pmod{\Delta},$$

since

$$p_{\sigma} \vec{a}_{\sigma} = \vec{0} \pmod{\Delta}.$$

Hence an upper bound (possibly not the least upper bound) on any cost, prior to all vectors being duplicated, will be

$$C' = \sum_{j=1}^n (p_j - 1) t_j, \text{ where } t_j = \min_j(C_j, 0).^5$$

If, at any point in the search for feasible solutions, C' is reached and no feasible solution is found prior to the generation of the next section, it can be assumed (and is by the program) that no feasible integer solution exists. This follows since it has been shown that all columns formed subsequently will duplicate prior columns which have already failed to produce a feasible solution.

Note that, given a large enough common denominator, the value of C' can become extremely high. However, the time and space requirements become prohibitively high at the same time, and efficient utilization of the program appears very unlikely for such problems. (This is discussed in detail in the section on Computational Results.)

⁵The definition of t_j allows a loose upper bound to be placed on costs for problems for which the continuous solution has one or more non-basic variables at their upper bounds (i.e., $C_j < 0$).

III. PROGRAMMING CONSIDERATIONS

Because all integer solutions to the congruence in (3) are systematically produced in order of increasing objective function value while searching for a solution, the original algorithm [1] necessarily would require considerable core storage in order to accommodate each intermediate section. Accordingly, several changes are introduced, the first of which modifies the procedure for identifying feasible solutions.

Rather than limiting a check for feasible solutions to the $\min C_j$ column at each iteration, the new section is checked at the time of its formation for possible integer solutions. This allows the peremptory discarding of all columns, formed subsequent to the computation of a possible solution, for which the C_j are greater than the one which produced the current feasible solution (i.e., if the column were retained and subsequently used to produce a feasible solution, the solution could not be optimal since the resulting objective function value would necessarily be greater than the current one; $d + C_r^*$). The utility of this particular modification depends entirely on how early in the iterative process a possible solution ("base" C_r^*) is found.

The algorithm provides for the elimination of duplicate columns only if the use of either would result in identical non-basic variable values. The program, however, allows the elimination of all duplicate columns in an initial search for an upper/lower bound (depending on minimization/maximization) on the optimal solution to the problem. The program then reruns the problem using the procedure originally specified by the algorithm in a search for the optimal solution. The bound placed on the optimal solution by the initial run of the program may be the optimal

solution itself; however, as will be explained, there is no guarantee that this will, in fact, be the case.

By definition, a duplicate column contains duplicate fractional parts as well as C_j values. Hence, it can be shown that the objective function values will always be identical when duplicate columns are used to produce integer solutions; but the basic variable values will be guaranteed to be identical only when their corresponding non-basic variable values are equal. Therefore, when a duplicate column is allowed to remain, based upon the fact that when selected it will produce non-basic variable values different from those of the original column, it will subsequently produce a section identical to the one produced by its "sister" column but with a different combination of non-basic variable values. When a column from one of the identical sections satisfies the $\beta_j = \beta_0$ requirement for an integer solution, the different non-basic variable values can produce a feasible solution in one case and a non-feasible solution in the other. Hence, if one column were to be peremptorily discarded, a feasible solution may be overlooked, and although the subsequent formation of duplicate columns may cause this solution to "reappear", it will continue to be lost due to its duplication. Test problem #3 demonstrates how this "course" duplicate column elimination method will run out of time and/or space prior to reaching an optimal solution. This is to be expected in a number of problems since an entire section necessarily takes longer to become totally marked, and hence its space does not become available for reuse until much later in the iterations, if at all. (This will be discussed in more detail later in this section). The "course" method, therefore, provides at the very least a bound on the optimal solution, although in all of the test problems

included here, these bounds were, in fact, optimal solutions. The following example should make clear, however, why this will not always be the case.

Assume that the following equivalent problem has been computed:

$$\begin{array}{ll} \text{minimize} & \frac{212}{8} + \frac{5}{8} x_2 + \frac{10}{8} x_3 \\ \text{subject to} & x_1 + \frac{3}{8} x_2 - \frac{2}{8} x_3 = \frac{4}{8} \\ & \frac{5}{8} x_2 - \frac{6}{8} x_3 + x_4 = \frac{12}{8} \end{array}$$

$$x_j \geq 0 \quad j = 1, 2, 3 ; 0 \leq x_4 \leq 2$$

Now form sections (1) and (2) according to the algorithm, noting that

$$\beta_0^T = (4, 4).$$

$$\begin{array}{cc} 2 & 3 \\ \hline 5 & 10 \\ 3 & 6 \\ 5 & 2 \\ \hline * & * \\ x_j=0 \end{array} \quad (1)$$

$$\begin{array}{cc} 2 & 3 \\ \hline 10 & 15 \\ 6 & 1 \\ 2 & 7 \\ \hline * & \\ x_2=1 \end{array} \quad (2)$$

Now form two new sections (a) and (b) first using Column 3, section (1) then Column 3, section (2).

2	3	
15	20	
1	4	
7	4	
$x_3=1$		(a)

2	3	
15	20	
1	4	
7	4	
$x_2=2$		(b)

Note that integer solutions may be found in columns 3 of (a) and 3 of (b). For the solution from (a), substitute $x_3 = 2$ into the equivalent constraint equations and get $x_1 = 1$ and $x_4 = 3$, an infeasible solution. For the solution from (b), substitute $x_2 = 2$ and $x_3 = 1$ into the equivalent constraint equations and get $x_1 = 0$ and $x_4 = 1$, a feasible solution.

Here, we see that if column 3 of (b) had been marked as being a duplicate, while column 3 of (a) had been retained, the solution would have been judged infeasible and all other solutions eliminated. This is, in fact, what happened in test problem number 9. It was not until the more strict "fine" duplicate column elimination criterial of the original algorithm were imposed that the "feasible" column was retained (this based upon the fact that, when chosen, it would produce different non-basic variables from its "sister" column) .

For the same reason that a feasible solution could have been bypassed in the example, one could also have been overlooked which would yield a lower objective function value than the current solution. For example,

consider the case where, prior to the formation of sections (a) and (b) above, a possible solution had been recorded which utilized the following column from a previously formed section:

$$(C_j \beta_j) = (28 \quad 4 \quad 4) .$$

It can be seen that the value of C_j here would yield a higher objective function value than either of the two columns previously discussed. The optimal solution would then be bypassed by virtue of the course duplicate column elimination criteria. However, should the program run out of time and/or space on the "fine" run prior to reaching the sections containing the columns above, we would at least have an upper bound on the minimum objective function value.

The two methods for duplicate column elimination are also made use of when the program is required to search for alternative optimal solutions. For the same reason that no optimal solution may be found on the first run, alternative solutions may also be bypassed. Hence, when the option to search for additional solutions is exercised, the program once again searches under both modes of duplicate column elimination. Time and space limitations may be binding here also, but given enough of both, the program will provide all alternative optimal solutions when required. In problems where two or more optimal solutions are found in the same section, all are recorded (whether or not alternative optima have been requested by the user) since no additional iterations will be required.

The principle space saving technique utilized by the program is one which makes section space available for newly formed sections once the columns of a previously formed section have all been marked. This is possible since the x_j values associated with the original section will

never be utilized; at no time will a column be reselected once it has been marked, either as a potential solution or as the min C_j column.

Several purely computer-related techniques were investigated in further attempts to decrease the core storage requirements of the program. Among these were the use of an overlay structure and the use of direct-access storage areas. Although an overlay structure⁶ has been incorporated in the final program, its use accounts for only a small saving of space. The use of the direct access devices did decrease storage requirements considerably, but computation time increased on the average by a factor of six; this due primarily to the fact that the problems requiring the most space also require a considerable number of iterations, and prohibitively high direct access retrieval times for each iteration make the use of this technique highly inefficient.

⁶Appendix B gives a brief discussion of the overlay technique utilized by the program.

IV. TEST PROBLEMS

Problems 1 through 9 were taken directly from a collection of twenty-five integer programming test problems compiled at Stanford University by John Haldi [4]. Of particular interest were problems 1, 3, 4, 5, 6, and 9. Upper bounds were computed for problems 1 and 3 using the "course" duplicate column elimination criteria, but the program ran out of space prior to reaching the optimal solution itself under the "fine" duplicate column elimination run. Problem 4 was a good test of how the program handles problems whose continuous solution includes non-basic variables at their upper bounds (i.e., the program allows negative C_j values as long as the upper bound constraints are maintained for the integer portion of the solution). Problem 9 provided an example of how the "course" duplicate column elimination run can bypass the optimal solution by eliminating (marking) all columns duplicating the one which produced an infeasible solution, regardless of their corresponding non-basic variable values.

Problems 5 and 6 are identical to problems 2 and 4 with the exception of one value in each. The Haldi paper gave optimal solutions for these two problems which did not satisfy the constraints as stated. However, with the modifications of problems 5 and 6, the answers given were both optimal and feasible.

Problem 10 is the example problem from the original paper by Dr. Greenberg [1], which demonstrates the use of the algorithm on bounded variable problems.

Problem 11 is a simple text book example [5] which depends heavily upon the algorithm modification requiring an immediate search for integer

solutions. This problem also appears in reference [8] to demonstrate an all integer algorithm by R. E. Gomory.

Problem 12 is an exercise from [9] which demonstrates the existence of a set of relatively small integer LP problems the solution of which can frequently be quite tedious and inefficient when employing R. E. Gomory's all integer algorithm [8]. Although it has been proven that this particular algorithm will converge to an integer solution in all cases, this convergence can be so slow as to require the use of an alternate all integer procedure such as the algorithm employed by the program presented here. (Note the relative ease with which the program handles solution of the problem).

Problem 13 was specifically developed to demonstrate how the program will handle a problem which has a feasible continuous solution but no feasible integer solution.

Except where specifically indicated, the variables are bounded below by zero and above by infinity and are all required to be integer.

A. SIX FIXED CHARGE PROBLEMS [4]:

$$\text{Maximize} \quad C^T X$$

$$\text{Subject to} \quad F\delta(X) + A^T(X) \leq b$$

$$X_i \geq 0$$

$$\delta(X) = \begin{cases} 0 & X_i = 0 \\ 1 & X_i > 0 \end{cases}$$

All problems are similar except for parameters a_1 , a_2 , and b_2 and all unsubscripted variables are vectors.

	F1	F2	A1	A2	A3	S1	S2	S3	S4
max	0	0	1	1	1	0	0	0	0

	F1	F2	A1	A2	A3	S1	S2	S3	S4	
subject to	2	3	1	2	2	1	0	0	0	b_1
	3	2	2	1	2	0	1	0	0	b_2
	$-a_1$	0	0	0	0	0	0	1	0	
	0	$-a_2$	0	0	0	0	0	0	1	

Problem	a_1	a_2	b_1	b_2
1	6	7	18	15
2	9	7	18	17
3	9	9	21	21
4	6	8	19	25
5	7	7	18	17
6	6	8	19	15

B. THREE IBM TEST PROBLEMS [4]:

Problems 7 and 8 are identical with the exception of b . (Note: (Note: slack variables must be added).

$$\text{Min } \vec{c}'\vec{x}$$

subject to

$$A\vec{x} \geq \vec{b}, x_j \geq 0$$

	x_1	x_2	x_3	x_4	x_5	x_6	x_7
min	1	1	1	1	1	1	1
subject to:	1	0	0	1	1	0	1
	0	1	0	1	0	1	1
	0	0	1	0	1	1	1
	1	1	0	0	1	1	0
	1	0	1	1	0	1	0
	0	1	1	1	1	0	0
	1	1	1	0	0	0	1

Problem	b_1	b_2	b_3	b_4	b_5	b_6	b_7
7	5	5	5	4	4	4	3
8	4	4	4	3	3	3	2

Problem 9:

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	b
min	13	15	15	11	0	0	0	
subject to:								
	4	5	3	6	-1	0	0	= 96
	20	21	17	12	0	-1	0	= 200
	11	12	12	7	0	0	-1	= 101

C. MISCELLANEOUS PROBLEMS:

Problem 10:

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	b
min	5	7	10	3	1	0	0	0	
subject to:									
	1	-3	5	1	-4	-1	0	0	= 2
	-2	6	-3	-2	2	0	-1	0	= 0
		-2	2	-1	-1	0	0	-1	= 1

$$0 \leq x_j \leq 1 \quad j = 1, \dots, 5$$

Problem 11:

	x_1	x_2	x_3	x_4	x_5	x_6	b
min	10	14	21	0	0	0	
subject to:							
	8	11	9	-1	0	0	= 12
	2	2	7	0	-1	0	= 14
	9	6	3	0	0	-1	= 10

Problem 12:

	x_1	x_2	x_3	x_4	b
min	-1	-4	0	0	
subject to:					
	2	4	1	0	= 7
	10	3	0	1	= 15

Problem 13:

	x_1	x_2	x_3	x_4	x_5	x_6	b
min	5	10	0	0	0	0	
subject to:							
	10	10	1	0	0	0	= 55
	10	10	0	-1	0	0	= 45
	-10	10	0	0	1	0	= 5
	10	-10	0	0	0	1	= 5

V. COMPUTATIONAL RESULTS

Two factors significantly affect the success or failure of the program as it processes each problem. One of these is the spread between the maximum and minimum coefficients in the objective function of the equivalent problem. Should this spread be wide enough, it can easily be demonstrated that the columns corresponding to the one with the high objective function coefficient in subsequent sections may require an extremely large number of sections prior to being marked as the minimum C_j . This will require that each section be provided "fresh" section space in core (i.e., there will be no space available for reuse since no section will become fully marked until late in the computation), and that the program runs the high risk of running out of space prior to reaching a solution.

Another obstacle to successful problem solution is the size of the maximum period for each section. As described previously, the higher the period, the more the distinct column vectors of fractional parts which can be formed by the algorithm; hence, the column vector which produces the optimal solution very likely will be generated only after a large number of sections. Table I is a compilation of computational results from five additional problems which the program failed to solve.⁷ The high section counts and corresponding high computation times make the use of an iterative scheme of the type employed by the algorithm very

⁷These problems are listed as numbers 5 through 9 of the first 10 appearing in [4] and are of interest here for two reasons: none was larger than six constraints and 11 variables, and all had relatively high common denominators.

TABLE I
Additional Test Problem Results

<u>PROBLEM</u>	<u>COMMON DENOMINATOR</u>	<u>SECTIONS</u> ¹⁰	<u>APPROXIMATE TIME (SECS)</u>
5 and 7 ⁹	19400	1744	651
		754	268
6 and 8 ⁹	32000	2591	969
		967	350
9	2000	1400	895
		878	486

<u>PROBLEM</u>	<u>MAX C_j WHEN PROGRAM STOPPED</u> ¹⁰	<u>SECTION #1</u>		<u>C_j REQUIRED TO PRODUCE INTEGER SOLUTIONS</u> ⁸
		<u>Min C_j</u>	<u>Max C_j</u>	
5 and 7	59750	4000	9000	244600
	30450			
6 and 8	118000	1300	5550	398000
	56000			
9	5600	400	800	6000
	4400			

⁸Computed using the answers supplied in [4].

⁹Both problems are identical with one exception: the first has bounded variables and the second introduces two additional constraints to produce the bounds.

¹⁰The first figure represents values for the problem under the "course" duplicate column elimination method; the second figure represents values when run under the "fine" method. All timing figures are exclusive of input/output times. All problems stopped because the limitation of 450 section spaces was exceeded.

unattractive economically, at least for those problems whose common denominators cannot be reduced into the hundreds.¹¹

In addition to the inefficiencies associated with problems with high common denominators, there also exists the possibility, albeit remote, of computer oriented round-off errors for these same problems. The round-off errors occur in the computation of the continuous solution basic variable values and equivalent problem coefficients and have the effect of changing a problem with a feasible integer solution to one with no feasible integer solution. This has occurred, however, in just two problems (listed in Table I), both of which had 32000 as their common denominators.¹² The particular error appeared as a miscalculation of $1/32000$ in one fractional part value. No other problem listed in Table I experienced such an error.¹³ Although these errors destroy any possibility of arriving at correct integer solutions, it would appear as if they occur only for extremely high common denominators and available data indicates that perhaps the iterative scheme should be abandoned for these problems in the first place.

In order to reduce total storage requirements by approximately one third, the size of the storage word for the fractional parts of the equivalent problem constraint coefficients and right-hand sides was reduced

¹¹By prior developments, the period is necessarily less than or equal to the common denominator; the highest period for the successful test problems was 320.

¹²The error was artificially corrected to produce the data for Table I.

¹³This fact was verified through the use of a computer program, developed by Dr. R. Shudde of the Naval Postgraduate School, which provides continuous results in integer format. The utilization of such a program will, of course, eliminate all possibility of round-off error.

by one-half. The particular procedure employed to effect this reduction is peculiar to the IBM System 360 and has the effect of limiting the values of equivalent problem common denominators to 32767; in light of the preceding discussion, this does not however appear to be too unreasonable a restriction.

TABLE II

Solutions

PROB- LEM	Optimal Objective Function Value	OPTIMAL VARIABLE COSTS													
		Subscripts													
		1	2	3	4	5	6	7	8	9	10	11	12	13	14
1 ¹⁴	7	0	0	0	0	7	4	1	0	0					
		0	0	0	0	8	2	1	0	0					
2	8	1	1	3	5	0	0	1	6	2					
		1	1	4	4	0	1	0	5	3					
		1	1	3	4	1	0	0	6	3					
3 ¹⁴	10	1	1	4	6	0	0	2	5	3					
		1	1	5	5	0	1	1	4	4					
		0	0	0	0	10	1	1	0	0					
4	11	1	0	6	0	5	1	0	0	0					
		1	0	5	0	6	0	0	1	0					
5	8	0	0	0	0	8	2	1	0	0					
		1	1	3	5	0	0	1	4	2					
		1	1	4	4	0	1	0	3	3					
		1	1	3	4	1	0	0	4	3					
6	8	0	1	0	8	0	0	5	0	0					
		0	1	0	7	1	0	4	0	1					
		0	1	0	6	2	0	3	0	2					
		0	1	0	5	3	0	2	0	3					
		0	1	0	4	4	0	1	0	4					
		1	1	2	6	0	0	0	4	2					
		0	1	0	3	5	0	0	0	5					

¹⁴Time/space limitations were reached prior to solving for all alternative optimal solutions.

TABLE II-- CONTINUED

PROB- LEM	Optimal Objective Function Value	OPTIMAL VARIABLE COSTS													
		Subscripts													
		1	2	3	4	5	6	7	8	9	10	11	12	13	14
7	8	1	0	0	2	2	1	2	2	0	0	0	0	0	0
		0	1	0	2	1	2	2	0	2	0	0	0	0	0
		1	1	0	1	2	2	1	0	0	0	2	0	0	0
		1	0	1	2	1	2	1	0	0	0	0	2	0	0
		0	1	1	2	2	1	1	0	0	0	0	0	2	0
		1	1	1	1	1	1	2	0	0	0	0	0	0	2
8	7	1	0	0	2	2	0	2	3	0	0	0	0	1	1
		1	1	0	1	1	1	2	1	1	0	1	0	0	2
		1	0	1	1	1	1	2	1	0	1	0	1	0	2
		1	1	1	1	1	0	2	1	0	0	0	0	1	3
		0	1	1	1	1	1	2	0	1	1	0	0	1	2
		1	1	1	1	0	1	2	0	1	0	0	1	0	3
		1	1	1	0	1	1	2	0	0	1	1	0	0	3
		1	1	1	1	1	1	1	0	0	0	1	1	1	2
		0	0	0	2	1	2	2	1	2	1	0	1	0	0
		0	1	0	2	1	1	2	1	2	0	0	0	1	1
		0	0	0	1	2	2	2	1	1	2	1	0	0	0
		0	0	1	1	2	1	2	1	0	2	0	0	1	1
		1	0	0	1	2	2	1	1	0	1	2	1	0	0
		1	1	0	1	2	1	1	1	0	0	2	0	1	1
		1	0	1	2	1	1	1	1	0	0	0	2	1	1
		0	1	1	2	2	0	1	1	0	0	0	0	3	1
		0	1	0	2	0	2	2	0	3	0	0	1	0	1
		0	0	1	1	1	2	2	0	1	2	0	1	0	1
		1	1	0	1	1	2	1	0	1	0	2	1	0	1
		0	1	0	1	2	2	1	0	1	1	2	0	1	0
		1	0	1	2	0	2	1	0	1	0	0	3	0	1
		0	0	1	2	1	2	1	0	1	1	0	2	1	0
		0	1	1	2	1	1	1	0	1	0	0	1	2	1
		0	0	1	0	2	2	2	0	0	3	1	0	0	1

TABLE II— CONTINUED

PROB- LEM	Optimal Objective Function Value	OPTIMAL VARIABLE COSTS													
		Subscripts													
		1	2	3	4	5	6	7	8	9	10	11	12	13	14
8	7	1	1	0	0	2	2	1	0	0	1	3	0	0	1
(Continued)		1	0	1	1	1	2	1	0	0	1	1	2	0	1
		1	1	0	1	2	2	0	0	0	0	3	1	1	0
		0	1	1	1	2	1	1	0	0	1	1	0	2	1
		1	0	1	2	1	2	0	0	0	0	1	3	1	0
		0	1	1	2	2	1	0	0	0	0	1	1	3	0
		0	0	0	2	2	1	2	2	1	1	0	0	1	0
		1	0	0	2	1	1	2	2	1	0	0	1	0	1
		1	0	0	1	2	1	2	2	0	1	1	0	0	1
		1	0	0	2	2	1	1	2	0	0	1	1	1	0
		1	0	0	2	1	2	1	1	1	0	1	2	0	0
		0	1	0	2	2	1	1	1	1	0	1	0	2	0
		0	0	1	2	2	1	1	1	0	1	0	1	2	0
		0	1	0	1	1	2	2	0	2	1	1	0	0	1
		0	1	0	2	1	2	1	0	2	0	1	1	1	0
		0	0	1	1	2	2	1	0	0	2	1	1	1	0
9	187	0	0	0	17	6	4	18							
10	17	0	1	1	0	0	0	3	0						
11	52	1	0	2	14	2	5								
12	5	5	1	1	1	2									
13		INFEASIBLE													

TABLE III
Sections Required and Computation Times

PROBLEM	COURSE ¹⁵		FINE ¹⁶	
	Sections ¹⁸	Time ¹⁷	Sections ¹⁸	Time ¹⁷
1	247	16.55	768+	272.06
2	253	21.75	769	287.48
3	340	28.20	967+	360.51
4	4	0.08	4	0.08
5	185	10.55	960	347.86
6	145	9.15	580	164.90
7	8	0.22	8	0.22
8	32	1.01	330	64.70
9	2000	416.40	412	65.60
10	5	0.06	5	0.06
11	39	0.49	39	0.49
12	3	0.01	3	0.01
13	INFEASIBLE			

¹⁵Refers to the modified duplicate column elimination method which will provide a bound for the optimal solution.

¹⁶Refers to the original duplicate column elimination method which ultimately provides the optimal solution.

¹⁷The the nearest 1/100 second; exclusive of input/output times and alternate optima requirements.

¹⁸Referred to as iterations in Table IV, this is new sections formed and not section space required; '+' means unsolved.

VI CONCLUSIONS

Table IV is a comparison of section¹⁹ requirements of the program with three IP programs available through SHARE [4]. Although in general the SHARE programs require considerably fewer iterations, the computer program presented here has several rather attractive benefits associated with its use. As discussed previously, all integer solutions to the equivalent knapsack problem are systematically produced in order of increasing objective function value; it is the associated iterative scheme of solution which will provide all existing alternate optima to integer LP problems. Also, as alluded to in the preceding discussion of test problem 12, the basic algorithm is a good substitute for Gomory's all integer algorithm in the solution of at least some integer problems. The computer program presented here does, however, have definite limitations.

As mentioned in the preceding section, the size of the period, and hence the common denominator, plays a major role in the program requirements for core and time; it is these requirements which ultimately restrict the size of the problem and the number of sections which the program will accommodate. Since the only test problems which failed to produce at least one optimal solution were those whose common denominators ranged from 2000 to 32000, the question arises: can this common denominator be approximated (i.e., reduced) to a point at which corresponding periods, and hence time and space requirements, are no longer prohibitively high and at the same time solutions do not become distorted? The ultimate utility of the program rests on the answer to this question.

¹⁹Program iterations refer to sections formed during the "fine" duplicate column elimination run; iterations for the three SHARE programs mean total pivot steps required to solve the problem.

TABLE IV
A Comparison of Iteration Requirements

<u>PROBLEM NUMBER</u>	<u>CONSTRAINTS</u>	<u>VARIABLES</u>	<u>PROGRAM</u>	<u>LIP1</u>	<u>IPOZ</u>	<u>IPO3</u>
1	4	9	768+	25	19	51
3	4	9	967+	22	17	80
5	4	9	960	21	18	58
6	4	9	580	17	29	35
					<u>IPMZ</u>	<u>IPM3</u>
7	7	14	8	11	7	5
8	7	14	330	30	10	11
9	3	7	412	63	2	28

APPENDIX A

Input Data and Output Formats

Input²⁰

- Card 1, cc 5: number of constraints
- cc 10: number of variables (including slack)
- cc 15: 1 if bounds are required
- cc 21-71: Problem title, if any ("STOP" after last card of last problem deck)
- cc 72: 1 if any existing alternative optimal solutions are to be listed
- Card Set 1: These are the constraint coefficients, A, in the form $A(I,J)$, where I is the constraint number, J the variable.
- cc 1: 1 if last card in the set
- cc 4: constraint number
- cc 7: variable number
- cc 12: coefficient value
- Card Set 2: These are the right hand sides, H, in the form $H(I)$, where I is the constraint number
- cc 1: 1 if last card in the set
- cc 4: constraint number
- cc 9: right hand side value

²⁰All input defaults to zero (bounds to zero and infinity). All data will be right adjusted to the card column (cc) indicated.

Card Set 3: These are the objective function costs, A , in the form $A(M,J)$, where J is the variable, M the number of constraints plus one (completed separately).

cc 1: 1 if last card in set

cc 4: variable number

cc 9: cost

Card Set 4: These are the upper bounds on the variables in the form $XUB(J)$, where J is the variable.

cc 1: 1 if last card in set

cc 4: variable number

cc 9: variable bound

Output

The first series of lines provides a review of the input data, in its original format. The continuous solution is then listed with its computation time. Next, all intermediate feasible solutions are listed with their individual computation times, iterations required and total section spaces required. (If alternate optima are available, they will appear at this point also.) The first section in the integer solution and the common denominator are then listed to provide a feel for the iteration requirements of the particular problem. Finally, the last optimal solution and its time and iteration requirements are listed.

APPENDIX B

Overlay Structure²¹

The program, with its present capacity, requires approximately 370 thousand bytes of storage in the IBM System/360-67 computer. By utilizing an overlay structure, this requirement can be reduced to approximately 360 thousand bytes. Although relatively small, this is as large a reduction as possible due to the high degree of interdependence among the subroutines in the program. Common blocks were specifically developed to take maximum advantage of the overlay structure.

The following cards, were placed directly behind the program in object deck form and were the product of the overlay tree of Figure 1.

OVERLAY ALPHA

INSERT LP, ELEM, PIVOT, MIN

OVERLAY ALPHA

INSERT INTPRG

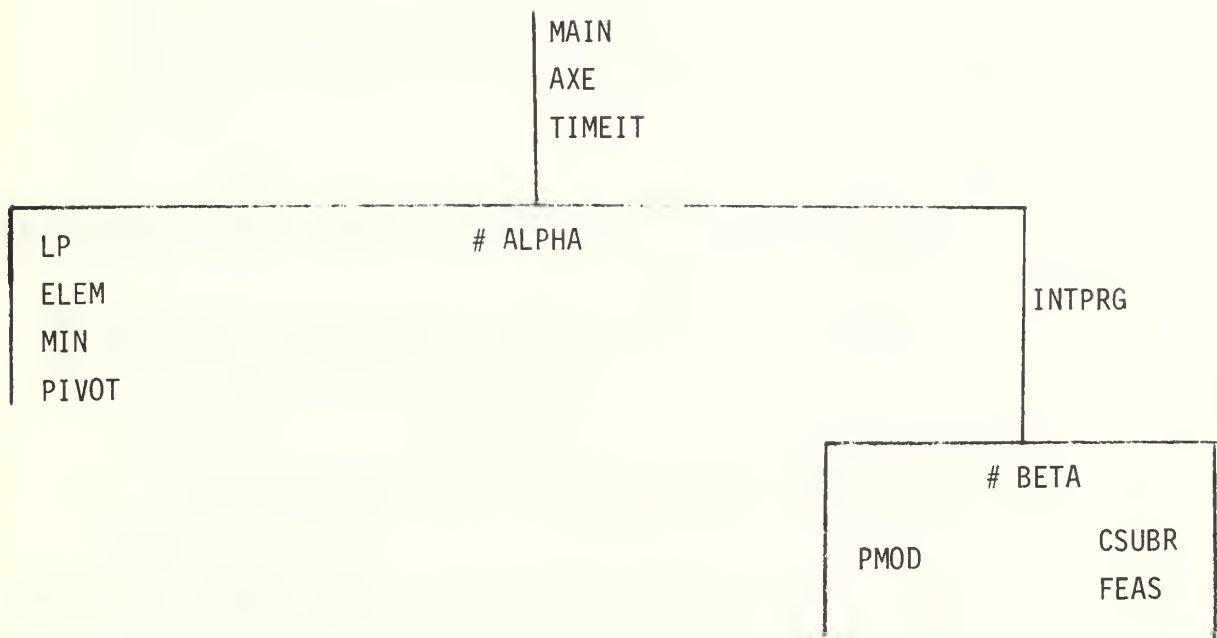
OVERLAY BETA

INSERT PMOD

OVERLAY BETA

INSERT CSUBR, FEAS

²¹References [10] and [11] were utilized extensively in the development of the overlay structure and should be consulted for a more detailed explanation of the techniques employed by the program.



THE OVERLAY TREE

Figure 1.

```

*****
*
*           A COMPUTER PROGRAM FOR THE SOLUTION
*
*           TO INTEGER LINEAR
*
*           PROGRAMMING PROBLEMS
*
*****

```

THE SUBPROGRAMS IMMEDIATELY FOLLOWING 'MAIN' (LP, ELEM, MIN, AXE, AND PIVOT) PROVIDE THE EQUIVALENT PROBLEM (I.E., THE CONTINUOUS SOLUTION TO THE ORIGINAL PROBLEM).

THE FINAL FOUR SUBPROGRAMS OPERATE ON THE EQUIVALENT PROBLEM, USING THE INTEGER ALGORITHM, TO PROVIDE AN INTEGER SOLUTION TO THE PROBLEM.

PROGRAM CAPACITY WILL ALLOW PROBLEMS OF UP TO 10 CONSTRAINTS AND 22 VARIABLES.

A TIMING ROUTINE IS PROVIDED AND WILL INDICATE INDIVIDUAL TIMES REQUIRED TO COMPUTE THE CONTINUOUS SOLUTION AND THE INTEGER SOLUTION.

ALL ALTERNATIVE OPTIMA WILL BE PROVIDED WHEN SETTING IALT = 1 (PUNCH '1' IN COL. 72 OF FIRST DATA CARD) CURRENT SPACE/TIME LIMITATIONS WILL NORMALLY PROVIDE A MINIMUM OF TWO OR THREE ALTERNATE SOLUTIONS.

PROGRAM TO ENTER DATA AND CALL LP

```

COMMON/ONE/XUB(24),X(24),IB(14),AX(14),XLB(24),H(14),
1A(14,24),XI(14)
COMMON/TWO/Y(14),B(14,14),ISOL(14),ASAVE(14,24),
1HSAVE(14)
COMMON/MM,N,M,JS,E,IR,AD,W,Z,IM,DI,ITR,XH,AH,HH,EE,RL,
1XT,XK,ID,BND,L,IW,ICLK,IGNAT,IOT,IOPP,IALT

```

PROGRAM REQUIRES DATA INPUT IN THE FORM OF A MINIMIZATION PROBLEM

IGNAT=0 MEANS MINIMIZE PROBLEM

IGNAT=1 MEANS MAXIMIZE PROBLEM

MM = NUMBER OF EQUATIONS

N = NUMBER OF VARIABLES

BND = 0 MEANS NO BOUNDS

BND = 1 MEANS BOUNDS

IALT = 1 MEANS ALTERNATIVE OPTIMA WILL BE PROVIDED

```

DIMENSION RUN(12)
INTEGER*4 RUN,STOPIT
DATA STOPIT/'STOP'/
INTEGER BND

```

8338 IOT=0

IOPP=0

READ(5,203)MM,N,BND,IGNAT,RUN,IALT

203 FORMAT(4I5,12A4,3X,11)

IF(RUN(1).EQ.STOPIT) STOP

```

204 WRITE(6,2(4)RUN
    FORMAT(1H1,29X13A4)
    WRITE (6,2)MM,N
    2 FORMAT(/74I5)
    M = MM+1
    DO 1 I=1,M
    H(I)=C.
    DO 1 J=1,N
    1 A(I,J)=C.
    DO 9 J=1,N
    XUB(J)=-1.
    9 XLB(J)=0.
    3 READ (5,4) IL,I,J,A(I,J)
    4 FORMAT(I1,2I3,F5.0)
    IF(IL.EQ.0)GO TO 3

C
C
C    H(I) ARE THE RIGHT HAND SIDES,H(M) = INITIAL Z
    5 READ (5,6) IL,I,H(I)
    6 FORMAT (I1,I3,F5.0)
    IF(IL.EQ.0)GO TO 5

C
C
C    A(M,J) ARE THE COSTS
    7 READ (5,6) IL,J,A(M,J)
    IF(IL.EQ.0)GO TO 7
    IF(BND.EQ.0)GO TO 88
    11 READ(5,12) IL,J,XUB(J)
    12 FORMAT (I1,I3,F5.0)
    IF(IL.EQ.0)GO TO 11

C
C
C    INITIAL INPUT VALUES ARE SAVED FOR USE ON SECOND
    ATTEMPT AT SOLUTION IF REQUIRED
    88 DO21I=1,MM
    DO21J=1,N
    21 ASAVE(I,J)=A(I,J)
    DO22I=1,M
    22 HSAVE(I)=H(I)
    DO23J=1,N
    23 ASAVE(M,J)=A(M,J)
    MSAVE=M
    MMSAV=MM
    NSAVE=N
    GOTO8
    55 N=NSAVE
    IOPP=0
    M=MSAVE
    MM=MMSAV
    DO24I=1,MM
    DO24J=1,N
    24 A(I,J)=ASAVE(I,J)
    DO25I=1,M
    25 H(I)=HSAVE(I)
    DO26J=1,N
    26 A(M,J)=ASAVE(M,J)
    WRITE(6,6000)
    6000 FORMAT('1')
    8 WRITE(6,10) ((A(I,J),J=1,N),I=1,MM)
    WRITE(6,10) (H(I),I=1,M)
    WRITE(6,10) (A(M,J),J=1,N)
    WRITE(6,10) (XUB(J),J=1,N)
    WRITE(6,10) (XLB(J),J=1,N)
    10 FORMAT(1H ,10F8.0)
    ICHK=0

C
C
C    TIMER STARTED FOR CONTINUOUS SOLUTION
    CALL TIMEIT( 0,TIMEX)
    CALLLP
    IF(ICHK.EQ.0)GOTO1063

```

```

C      TIMER STARTED FOR INTEGER SOLUTION
C
      CALL TIMEIT( 0,TIMEX)
      CALL INTPRG
      IF(IOT.EQ.1.AND.IOPP.EQ.1)GOTO222
      IF(IOT.EQ.1.AND.IOPP.NE.1)GOTO55
      GOTO1063
222  WRITE(6,223)
223  FORMAT(///' ABOVE SOLUTION(S) REPRESENT UPPER/LOWER BO
      1 OPTIMAL SOLUTION..PROBLEM NOW RERUN USING TOTAL DUP.
      WRITE(6,224)
224  FORMAT(/' IF THIS METHOD FAILS OR IS INCOMPLETE DUE TO
      1 LIMITATIONS, THIS/THESE BOUNDS IS/ARE THE BEST APPROXI
      WRITE(6,225)
225  FORMAT(/' TO A SOLUTION WHICH THE PROGRAM CAN COMPUTE.
      GOTO55
1063 WRITE(6,14)
      14 FORMAT(1H0,10X, ' HAVE FINISHED ')
      GOTO8888
      END

```

SUBROUTINE LP

C
C

```

COMMON/ONE/XUB(24),X(24),IB(14),AX(14),XLB(24),H(14),
1A(14,24),XI(14)
COMMON/TWO/Y(14),B(14,14),ISOL(14),ASAVE(14,24),
1HSAVE(14)
COMMON MM,N,M,JS,E,IR,AD,W,Z,IM,DI,ITR,XH,AH,HH,EE,RL,
1XT,XK,ID,BND,L,IW,ICLK,IGNAT,IOT,IOPP,IALT
IW=0
Z=H(M)
L=0
M=MM+2
ID=0
DO 3 J=1,MM
DO 2 I=1,M
2 B(I,J)=0.
B(M,J)=-1.
3 B(J,J)=1.
XI(J)=+1. MEANS INCREASE
XI(J)=-1. MEANS DECREASE
XI(J)=2. MEANS BASIC
XI(J)=0. MEANS NONBASIC
DO 24 J=1,N
XI(J)=0.
24 X(J)=0.
DO 4 I=1,MM
Y(I)=H(I)
4 IB(I)=-I
AD=1.
DI=1.
E=.5
W=0.
DO 5 J=1,N
A(M,J)=0.
DO 5 I=1,MM
5 A(M,J)=A(M,J)-A(I,J)
DO 6 I=1,MM
6 W=W+H(I)
ITR=0
ITER=0
8 JS=0
DO 9 J=1,N
IF(XI(J).EQ.2.) GO TO 9
D=0.
DO 7 K=1,MM
7 D=D+B(M,K)*A(K,J)
IF(W.EQ.0.) D=D+A(M,J)
IF(ABS(D).LE.E)GO TO 9
IF(D)15,9,16
15 IF(XUB(J).EQ.-1.)GO TO 19
IF(ABS(XUB(J)-X(J)).LT.E)GO TO 9
INCREASE
19 XI(J)=1.
GO TO 17
16 IF(ABS(X(J)-XLB(J)).LT.E)GO TO 9
DECREASE
XI(J)=-1.
17 IF(JS.GT.C)GO TO 10
61 JS=J
DJS=D*XI(J)
GO TO 9
10 IF(DJS-D*XI(J).GE.E)GO TO 61
9 CONTINUE
IF(JS.EQ.C)GO TO 101
221 CALLELEM
ITER=ITER+1
IF(IR.GT.0)GO TO 36
IF(BND.EQ.0)GO TO 33
IF(XH.EQ.1000000.)GO TO 33

```

```

36 DO 38 I=1,MM
   IF(IB(I))39,528,40
39 Y(I)=Y(I)-AX(I)*XH*XI(JS)
   GO TO 38
40 K=IB(I)
   X(K)=X(K)-AX(I)*XH*XI(JS)
38 CONTINUE
   X(JS)=X(JS)+XH*XI(JS)
   IF(W.EQ.0.)GO TO 62
   W=W+AX(M)*XH*XI(JS)
62 Z=Z+AX(MM+1)*XH*XI(JS)
   IF(IR)528,50,44
44 CALL PIVOT
50 IF(W.GT.E)GO TO 8
   IF(IW.EQ.1)GO TO 8
   W=0.
   M=M-1
   IW=1
   GO TO 8
33 CALLTIMEIT(-1,TIMEX)
   WRITE(6,542)TIMEX
542 FORMAT(//' TIME TO COMPUTE CONTINUOUS SOLUTION = ',
1-6PF15.6,' SECONDS.')
   WRITE(6,704)
704 FORMAT(1H0,10X,18HUNBOUNDED SOLUTION)
   GO TO 528
101 IF(W.GT.E)GO TO 35
   CALLTIMEIT(-1,TIMEX)
   WRITE(6,542)TIMEX
   WRITE(6,204)
204 FORMAT(1HC,5X,6HRESULT)
   DO 31 J=1,N
   IF(X(J).LT.E)GO TO 31
   WRITE(6,200)J,X(J)
200 FORMAT(1H ,5X,2HX(,12,2H)=,F10.4)
31 CONTINUE
   Z=SIGN(AINT(DI*ABS(Z)+.5)/DI,Z)
   WRITE(6,527)Z
527 FORMAT(1HC,5X,2HZ=,F10.5)
C HAVE CONTINUOUS SOLUTION
   IF(DI-1.)601,601,603
601 CALLTIMEIT(-1,TIMEX)
   WRITE(6,542)TIMEX
   WRITE(6,701)
701 FORMAT(1HC,5X,13HHAVE INTEGERS)
   DO45J=1,N
45 ISOL(J)=X(J)+.5
   DO47J=1,N
   IF(ISOL(J))48,47,48
48 WRITE(6,49)J,ISOL(J)
49 FORMAT(1HC,5X,2HX(,12,2H)=,18)
47 CONTINUE
   GOTO528
603 IF(ABS(ABS(Z)-AINT(ABS(Z)+E)).GE.E)GO TO 6032
   DO 637 I=1,MM
   IF(IB(I))637,637,83
83 J=IB(I)
   IF(ABS(X(J)-AINT(X(J)+E))-E)637,637,6032
637 CONTINUE
C HAVE ALL INTEGERS
   WRITE(6,701)
   GOTO528
35 CALLTIMEIT(-1,TIMEX)
   WRITE(6,542)TIMEX
   WRITE(6,205)
205 FORMAT(1HC,' INFEASIBLE ')
   GOTO528
6032 ICHK=1
   WRITE(6,703)
703 FORMAT(' HAVE FRACTIONS')
528 RETURN
END

```

SUBROUTINEELEM

C
C

```

COMMON/ONE/XUB(24),X(24),IB(14),AX(14),XLB(24),H(14),
1A(14,24),XI(14)
COMMON/TWO/Y(14),B(14,14),ISOL(14),ASAVE(14,24),
1HSAVE(14)
COMMON MM,N,M,JS,E,IR,AD,W,Z,IM,DI,ITR,XH,AH,HH,EE,RL,
1XT,XK,ID,BND,L,IW,ICLK,IGNAT,IOT,IOPP,IALT
IR=0
CALLAXE
EE=E
IF(BND.EQ.0.)GO TO 12
IF(XI(JS).EQ.1.)GO TO 1
GO TO 2
1 XH=XUB(JS)-X(JS)
IF(XUB(JS).EQ.-1.)XH=1000000.
GO TO 12
2 XH=X(JS)-XLB(JS)
DO 25 I=1,MM
IF(BND.GT.0.)GO TO 810
IF(AX(I).LE.E)GO TO 25
AH=AX(I)
HH=H(I)
IF(IR)25,811,812
811 IR=I
XH=HH/AH
EE=E/AH
GO TO 25
812 CALL MIN (I)
GO TO 25
810 IF(ABS(AX(I)).LT.E)GO TO 25
IF(XI(JS).EQ.1.)GO TO 8
C DECREASE
IF(IB(I))3,25,4
4 K=IB(I)
IF(AX(I))5,25,6
6 IF(XUB(K).EQ.-1.)GO TO 25
HH=XUB(K)-X(K)
AH=-AX(I)*XI(JS)
CALL MIN (I)
GO TO 25
5 HH=X(K)-XLB(K)
AH=AX(I)*XI(JS)
CALL MIN (I)
GO TO 25
3 IF(AX(I))15,25,16
16 HH=Y(I)
AH=-AX(I)*XI(JS)
CALL MIN (I)
GO TO 25
15 HH=Y(I)
AH=AX(I)*XI(JS)
CALL MIN (I)
GO TO 25
C INCREASE
8 IF(IB(I))30,25,40
40 K=IB(I)
IF(AX(I))6,25,5
30 IF(AX(I))16,25,15
25 CONTINUE
RETURN
END

```

SUBROUTINE MIN (I)

C
C

```

COMMON/ONE/XUB(24),X(24),IB(14),AX(14),XLB(24),H(14),
1A(14,24),XI(14)
COMMON MM,N,M,JS,E,IR,AD,W,Z,IM,DI,ITR,XH,AH,HH,EE,RL,
1XT,XK,ID,BND,L,IW,ICLK,IGNAT,IOT,IOPP,IALT
IF (ABS (HH-AH*XH)-EE)70,70,2
2 IF (HH-AH*XH)24,70,25
24 XH=HH/AH
3 IR=I
EE=E/AH
GO TO 25
70 IF (IB(I))74,25,73
73 IF (IB(IR))25,25,4
74 IF (IB(IR))4,25,3
4 IF (ABS (AX (IR))-ABS (AX (I)))25,25,3
25 RETURN
END

```

SUBROUTINE AXE

```

COMMON/ONE/XUB(24),X(24),IB(14),AX(14),XLB(24),H(14),
1A(14,24),XI(14)
COMMON/TWO/Y(14),B(14,14),ISOL(14),ASAVE(14,24),
1HSAVE(14)
COMMON MM,N,M,JS,E,IR,AD,W,Z,IM,DI,ITR,XH,AH,HH,EE,RL,
1XT,XK,ID,BND,L,IW,ICLK,IGNAT,IOT,IOPP,IALT
DO 21 I=1,M
AX(I)=0.
DO 21 K=1,MM
AX(I)=AX(I)+B(I,K)*A(K,JS)
21 CONTINUE
AX(MM+1)=AX(MM+1)+A(MM+1,JS)
22 RETURN
END

```

SUBROUTINE PIVOT

C
C

```

COMMON/ONE/XUB(24),X(24),IB(14),AX(14),XLB(24),H(14),
1A(14,24),XI(14)
COMMON/TWO/Y(14),B(14,14),ISOL(14),ASAVE(14,24),
1HSAVE(14)
COMMON MM,N,M,JS,E,IR,AD,W,Z,IM,DI,ITR,XH,AH,HH,EE,RL,
1XT,XK,ID,BND,L,IW,ICLK,IGNAT,IOT,IOPP,IALT
AD=AD*ABS(AX(IR))
DI=AIN(T(DI*ABS(AX(IR))+.5)
84 IF(IB(IR))1,1,2
2 K=IB(IR)
XI(K)=0.
1 IB(IR)=JS
XI(JS)=2.
E=.5/DI
DO 29 I=1,M
IF(I.EQ.IR)GO TO 29
P=-AX(I)/AX(IR)
H(I)=H(I)+P*H(IR)
DO 31 J=1,MM
31 B(I,J)=B(I,J)+P*B(IR,J)
29 CONTINUE
P=1./AX(IR)
H(IR)=H(IR)*P
DO 33 J=1,MM
33 B(IR,J)=B(IR,J)*P
DO 90 K=25,600,25
IF(K-ITR)90,91,95
90 CONTINUE
GO TO 95
91 DO 92 I=1,M
H(I)=SIGN(AINT(DI*ABS(H(I))+.5)/DI,H(I))
DO 92 J=1,MM
92 B(I,J)=SIGN(AINT(DI*ABS(B(I,J))+.5)/DI,B(I,J))
95 ITR=ITR+1
RETURN
END

```

SUBROUTINE INTPRG

```
COMMON/ONE/XUB(24),X(24),IB(14),AX(14),XLB(24),H(14),
1A(14,24),XI(14)
```

VARIABLES

AT MATRIX

ROW 1 = EQUIVALENT OBJECTIVE FUNCTION
COEFFICIENTS

ROW 2 = EQUIVALENT CONSTRAINT COEFFICIENTS

LAST COLUMN = EQUIVALENT RIGHT HAND SIDES

IX MATRIX

ROW 1 = NON-BASIC VARIABLE INDICES

ROW 2 = DUMMY (ORIGINALLY OBJECTIVE FUNCTION
NUMERATORS)

ROW 3(FF) = CONSTRAINT COEFFICIENT FRACTIONAL
PARTS

LAST ROW = MODIFIED NON-BASIC VARIABLE VALUES

IXX = OBJECTIVE FUNCTION NUMERATORS

IOPP=1 MEANS OPTIMAL SOLUTION REACHED

LIP = NUMBER OF POSSIBLE SOLUTIONS RECORDED

LT = ITERATION CURRENTLY BEING PROCESSED

IOT=1 RESULTS IN A RERUN OF THE PROBLEM AFTER
EITHER 450 SECTION SPACES HAVE BEEN USED OR
2000 ITERATIONS HAVE BEEN PERFORMED UNDER
THE MODIFIED DUPLICATE COLUMN ELIMINATION
RULE

IOT=2 MEANS THAT AFTER BOTH RUNS, SPACE AND
ITERATION LIMITS HAVE BEEN EXCEEDED AND THE
PROGRAM STOPS.

```
COMMON MM,N,M,JS,E,IR,AD,W,Z,IM,DI,ITR,XH,AH,HH,EE,RL,
1XT,XK,ID,BND,L,IW,ICLK,IGNAT,IOT,IOPP,IALT
```

IPP=1 VALUE OF CURRENT MIN C(J) PRODUCING A
POSSIBLE SOLUTION

IBJAY= OLD MIN C(J) TO BE RESTORED TO ICJAY IF NEW
SOLUTION TRIED IS INFEASIBLE
OR ITERATIONS EXCEED 2000, AND THE PROBLEM
IS THE RERUN USING A COMPLETE DUPLICATE
CHECK

NB(I)= NON-BASIC VARIABLE BOUNDS

KB(I)=BASIC VARIABLE BOUNDS

KBVAR(I)= BASIC VARIABLE INDICES

LK =COMMON DENOMINATOR FOR MODIFIED COEFFICIENT

IG MATRIX

OUTPUT OF PMOD - FRACTIONAL PARTS OF AT MATRIX
COMPONENTS

NBV= NUMBER OF NON-BASIC VARIABLES

IBV= NUMBER OF BASIC VARIABLES

N= NUMBER OF VARIABLES

```
COMMON/FOUR/LT,IA,JA,IT,NBV,LK,IPP,IAT,JAT,ICJAY,IOP,
1IBJAY,YLK,LKTT,LAP,KAT,IBV,LIP,IOU,MCR,ISUM,ICOST
1,JUMP,LOKP,LOKPT,NCJAY,MCJAY
```

M= NUMBER OF BASIC VARIABLES +1

MM= NUMBER OF CONSTRAINTS

MB(I)= NON-BASIC VARIABLE INDICES (VALUES UNVARIED

KEY=1 MEANS A POSSIBLE SOLUTION HAS BEEN FOUND

IOP=1 MEANS A POSSIBLE SOLUTION WAS FOUND
INFEASIBLE AND SEARCH MUST CONTINUE

LAP=1 MEANS A TOTALLY MARKED SECTION IS TO PROVIDE
THE SPACE FOR A NEW SECTION

KAT= THE VALUE WHICH SAVES THE TOTAL NUMBER OF
SECTIONS AT THE TIME THE SECTION SPACE IS TO
BE RE-USED

LKTT= ITERATION COUNT

NB(J)=NON-BASIC VARIABLE BOUNDS

JUMP=1 MEANS THERE IS NO FEASIBLE SOLUTION TO THE

```

C      PROBLEM
C      ISUM = COMPUTED MAXIMUM POSSIBLE C(J), AFTER WHICH
C      ALL COLUMNS OF FRACTIONAL PARTS WILL BE
C      DUPLICATES
C
C      COMMON/THREE/KBVAR(10),KB(10),MX(13),MB(23),NB(10),
1    1AT(23,23),JX(13),AMX(13),MOX(22),IP(13,23),IXX(23,450)
1    1,IIIP(22),NG(13),IX(13,23,450),IG(23,23)
      INTEGER*2 IX,IG
C
C      VALUES FROM CONTINUOUS SOLUTION ARE IDENTIFIED AND
C      ORDERED
C
C      NON-BASIC VARIABLE INDICIES AND BOUNDS
C
      NCTS=N-MM+1
      NOT=0
      DO1311 J=1,N
      XTE=X(J)-XUB(J)
      YTE=ABS(XTE)
      IF(X(J).GT.E.AND.YTE.GT.E)GOTO1311
      NOT=NOT+1
      IX(1,NOT,1)=J
      IF(XUB(J).EQ.-1)GOTO1322
      NB(NOT)=XUB(J)
      GOTO1311
1322 NB(NOT)=999999
1311 CONTINUE
      IBV=MM
      KIBV=IBV+1
      AT(1,NOTS)=-H(KIBV)
      NBV=N-IBV
      JA=NBV+1
      IA=IBV+3
C
C      NON-BASIC VARIABLE VALUES INITIALIZED
C
      DO156 I=1,JA
156  IX(IA,I,1)=0
      NOTP=NBV+1
      IX(1,NOTP,1)=0
      M=MM+1
      LK=DI
C
C      DIAGNOSTIC MESSAGE PRODUCED WHEN THE COEFFICIENT
C      MATRIX COMMON DENOMINATOR EXCEEDS THE CAPACITY FOR
C      INTEGER*2 (32767), A SPACE SAVING DEVICE TO
C      INCREASE PROGRAM CAPACITY. I.E., THE FRACTIONAL
C      PARTS OF THE COEFFICIENTS (OBJECTIVE FUNCTION
C      COEFFICIENTS ARE HANDLED SEPARATELY) COULD EXCEED
C      INTEGER*2 CAPACITY.
C
      IF(LK.LE.32767)GOTO1156
      WRITE(6,3333)LK
3333  FORMAT(//)' PROGRAM TERMINATED DUE TO POSSIBLE OVERFL
1    1Y A HIGH COEFF. MAT. COMMON DENOMINATOR. (LK = ',I6,')
      STOP
C
1156  YLK=0.0
      JSS=0
      DO9995 JS=1,N
      CALL AXE
      DO9997 I=1,IBV
      IF(JS.EQ.IB(I))GOTO9998
9997  CONTINUE
      JSS=JSS+1
C
C      BASIC VARIABLE VALUES

```

```

C      DO9996 JST=2,M
      JSTT=JST-1
9996 AT(JST,JSS)=AX(JSTT)
C
C      OPTIMAL CONTINUOUS SOLUTION (OR EQUIVALENT VALUE WHEN
C      A NON-BASIC VARIABLE IS AT ITS UPPER BOUND)
C
      AT(1,JSS)=AX(M)
C
      GOTO9995
9998 NGO=0
      DO9991 I=1,MM
      AXB=AX(I)-1.
9991 IF (ABS(AXB).LE.E) NGO=I
9992 KBVAR(NGO)=JS
      NGOO=NGO+1
      AT(NGOO,N0TP)=H(NGO)
      IF(XUB(JS).EQ.-1) GOTO55
      KB(NGO)=XUB(JS)
      GOTO9995
55 KB(NGO)=999999
9995 CONTINUE
      N=N-M+2
503 IAT=IA-1
      JAT=JA-1
668 DC33 I=1,N
33 MB(I)=IX(1,I,1)
      ICJAY=2147483646
      IBJAY=2147483646
      MCJAY=2147483646
      NCJAY=2147483646
C
C      COMPUTATION OF COEFFICIENT MATRIX FRACTIONAL PARTS
C      (I.E., MOD LK)
C
      CALL PMOD(M,N)
C
      DO 6 I=2,M
      DO 6 J=1,N
      NO=I+1
6 IX(NO,J,1)=IG(I,J)
C
C      COMPUTATION OF NUMERATOR VALUES OF EQUIVALENT
C      OBJECTIVE FUNCTION
C
      LOKP=0
      LOKPT=2147483646
      DO 8 I=1,JA
      TLK=LK
      TALK=AT(1,I)*TLK
      BALK=TALK+.5
      IF(AT(1,I).LT.0) BALK=TALK-.5
      IX(2,I,1)=0
      IXX(I,1)=BALK
      IF(IXX(I,1).LE.LOKPT) LOKPT=IXX(I,1)
8 IF(LOKPT.LT.0) LOKP=1
C
C      MAXIMUM POSSIBLE C(J) IS COMPUTED FOR USE IN
C      IDENTIFYING INTEGER PROBLEMS WITH NO FEASIBLE
C      INTEGER SOLUTION.
C
      ALOCK=LK-1
      ISQ=SQRT(ALOCK)
      DO667 J=1,JAT
      DO670 I=3,IAT
      DO6668 L=1,ISQ
      LWW=L-1
      LW=ISQ-LWW
      PLW=LW
      APR=IX(I,J,1)

```

```

        PLK=LK
        ALL=APR/PLW
        ALLL=PLK/PLW
        ILL=ALL
        ILLL=ALLL
        CLL=ILL
        CLLL=ILLL
        BLL=ALL-CLL
        BLLL=ALLL-CLLL
        IF(BLL.EQ.0..AND.BLLL.EQ.0.)GOTO 670
6668 CONTINUE
670 NG(I)=LW
    KG=999999
    DO669K=3,IAT
669 IF(NG(K).LE.KG)KG=NG(K)
667 IIIP(J)=LK/KG
    ISUM=0
    DO671J=1,JAT
    IIIIP=IIIP(J)-1
    IXXTT=IXX(J,1)
    IF(IXX(J,1).LT.0)IXXTT=0
    ICOST=IIIIP*IXXTT
671 ISUM=ISUM+ICOST
    JUMP=0
C
    LIP=0
    LT=1
    IOP=0
    LAP=0
    KAT=C
    LKTT=1
    IOU=0
45 IPP=0
    LP=LT
    DO11LODE=LP,3000
    IF(IPP.EQ.1)GOTO200
C
C    SECTIONS ARE OPERATED ON TO IDENTIFY POSSIBLE
C    SOLUTIONS AND FORM NEW SECTIONS IF REQUIRED
C
    CALL CSUBR(N,IR,IGNAT,IOT,IOPP,IALT)
C
    IF(JUMP.EQ.1)GOTO173
    IF(IOPP.EQ.1)GOTO311
    IF(IOT.EQ.1.AND.IOU.EQ.1)GOTO504
    IF(IOT.EQ.2)GOTO1056
11 CONTINUE
C
C    POSSIBLE SOLUTIONS CHECKED FOR FEASIBILITY AND EITHER
C    RECORDED OR DISCARDED
C
200 CALL FEAS (IR,M,N,IGNAT)
C
    GOTO45
C
C    FINAL SOLUTION AND INITIAL SECTION ARE PRINTED
C
311 CALLTIMEIT(-1,TIMEX)
    IF(IALT.EQ.1.AND.MCR.GE.ICJAY.AND.IOT.EQ.2)WRITE(6,889
889 FORMAT('///// ALL EXISTING ALTERNATE OPTIMA ARE INCLUDE
1DIATE SOLNS. FOR WHICH OBJ.FN. VALUES = FINAL SOLN VAL
    IF(IOT.EQ.1.AND.IALT.EQ.1)GOTO505
1056 WRITE(6,102)
102 FORMAT('1'/////
1    SECTION #1')
    DO1057I=1,JAT
1057 IX(1,I,1)=MB(I)
    DO104I=1,IA
    IF(I.EQ.2)GOTO144
    WRITE(6,103)(IX(I,J,1),J=1,JA)
    GOTO104
144 WRITE(6,103)(IXX(JO,1),JO=1,JA)

```

```

104 CONTINUE
103 FORMAT(//1X10I12)
    WRITE(6,101)LK
101 FORMAT(////' COMMON DENOMINATOR USED TO COMPUTE FRA
    1S = ',I7)
    IF(JUMP.EQ.1)GOTO504
    IF(ICT.EQ.2.AND.LIP.EQ.0)GOTO1050
    WRITE(6,400)
400 FORMAT('1'////'
    1 OPTIMAL SOLUTION')
    WRITE(6,401)LKTT,LT
401 FORMAT(////' TOTAL INTEGER ITERATIONS = ',I10,'. TOTAL
    1CES REQUIRED = ',I4,' OF 450 AVAILABLE.')
    WRITE(6,3004)
3004 FORMAT(//' NON BASIC VARIABLES INTEGER SOLUTION VALUES
    DO212I=1,JAT
    WRITE(6,3002)MB(I),MOX(I)
3002 FORMAT(/' X( ',I2,' )=',I6)
212 CONTINUE
    DO66I=1,IBV
    LZ=I+2
66 WRITE(6,300)KBVAR(I),MX(LZ)
300 FORMAT(//' INTEGER SOLN. VALUE FOR BASIC VARIABLE, X('
    1)
    WRITE(6,301) MX(2)
301 FORMAT(////' OPTIMAL OBJECTIVE FUNCTION INTEGER SOLUTI
505 WRITE(6,542)TIMEX
542 FORMAT(//' COMPUTATION TIME SINCE LAST POSSIBLE INTEGE
    1',-6PF15.6,' SECONDS.')
    GOTO504
1050 WRITE(6,1055)LKTT
1055 FORMAT(//' SECTIONS REQUIRED EXCEEDS 450. TOTAL ITERAT
    1R = ',I10)
    GOTO504
173 WRITE(6,174)ISUM,LKTT
174 FORMAT(////' UPPER BOUND ON COST,',I10,' REACHED..ALL
    1LUMNS FORMED..NO FEASIBLE INTEGER SOLUTION EXISTS..ITE
    1I6)
    GOTO311
504 RETURN
    END

```

C
C
C
C
C
C
C
C
C

SUBROUTINE PMOD(M,N)

PROGRAM TO COMPUTE FRACTIONAL PARTS OF THE EQUIVALENT
PROBLEM CONSTRAINT EQUATIONS USING THE COMMON
DENOMINATOR, LK.

ELEMENTARY ARITHMETIC PROCESSES AND ROUNDING
TECHNIQUES UTILIZED

```

COMMON/ONE/XUB(24),X(24),IB(14),AX(14),XLB(24),H(14),
1A(14,24),XI(14)
COMMON/THREE/KBVAR(10),KB(10),MX(13),MB(23),NB(10),
1AT(23,23),JX(13),AMX(13),MOX(22),IP(13,23),IXX(23,450)
1,IIIP(22),NG(13),IX(13,23,450),IG(23,23)
INTEGER*2 IX,IG
COMMON/FOUR/LT,IA,JA,IT,NBV,LK,IPP,IAT,JAT,ICJAY,IOP,
1IBJAY,YLK,LKTT,LAP,KAT,IBV,LIP,IOU,MCR,ISUM,ICOST
1,JUMP,LOKP,LOKPT,NCJAY,MCJAY
DO 12I=2,M
DO 12J=1,N
IF(AT(I,J).EQ.0.0)GOTO10
GOTO11
10 G=0.0
GOTO1
11 IF(AT(I,J).GT.0.0)GOTO2
B=AT(I,J)-.5
GOTO3
2 B=AT(I,J)+.5
3 IZ=B
R=IZ
IF(ABS(AT(I,J)).LE.ABS(R))IZ=AT(I,J)
R=IZ
IF(AT(I,J).LT.0.0)GOTO4
EX=AT(I,J)-R
GOTO5
4 D=AT(I,J)+ABS(R)
IF(D.EQ.0.0)GOTO8
GOTO9
8 G=0.0
GOTO1
9 EX=1.0+D
5 F=EX*LK
G=F+.5
1 IG(I,J)=G
IF(IG(I,J).EQ.LK)IG(I,J)=0
12 CONTINUE
RETURN
END

```

```

C
C
C
C
C
SUBROUTINE CSUBR(N,IR,IGNAT,IOT,IOPP,IALT)

PROGRAM TO DETERMINE MINIMUM C(J), TEST FOR POSSIBLE
SOLUTIONS, AND CONSTRUCT NEW SECTIONS

COMMON/ONE/XUB(24),X(24),IB(14),AX(14),XLB(24),H(14),
1A(14,24),XI(14)
COMMON/THREE/KBVAR(10),KB(10),MX(13),MB(23),NB(10),
1AT(23,23),JX(13),AMX(13),MOX(22),IP(13,23),IXX(23,450)
1,IIP(22),NG(13),IX(13,23,450),IG(23,23)
INTEGER*2 IX,IG
COMMON/FOUR/LT,IA,JA,IT,NBV,LK,IPP,IAT,JAT,ICJAY,IOP,
1IBJAY,YLK,LKTT,LAP,KAT,IBV,LIP,IOU,MCR,ISUM,ICOST
1,JUMP,LOKP,LOKPT,NCJAY,MCJAY
KEY=0

C
C
C
C
IF(IOP.EQ.0)GOTO914

A POSSIBLE SOLUTION WAS FOUND INFEASIBLE AND THE
PREVIOUS MIN C(J) MUST BE RESTORED

IX(1,IR,IT)=0
ICJAY=IBJAY
IOP=0
NCJAY=MCJAY

C
C
C
NEWLY FORMED SECTION IS CHECKED FOR POSSIBLE SOLUTIONS

914 DO88J=1,JAT
IF(IX(1,J,LT).LE.0)GOTO88
IF(IXX(J,LT).GT.NCJAY)GOTO89
IF(IXX(J,LT).GT.ICJAY.AND.IXX(J,LT).LE.NCJAY.AND.LOKP.
1EQ.1)GOTO88
IF(IXX(J,LT).GT.ICJAY.AND.LOKP.NE.1)GOTO89
DO8I=3,IAT
RLK=(IX(I,J,LT)-IX(I,JA,1))
IF(ABS(RLK).LE.YLK)GOTO8
GOTO88
8 CONTINUE
IBJAY=ICJAY
MCJAY=NCJAY
ICJAY=IXX(J,LT)
NCJAY=IXX(J,LT)
IF(LOKP.EQ.1)NCJAY=IXX(J,LT)-LOKPT
IR=J
KEY=1
GOTO188
89 IX(1,J,LT)=0
38 CONTINUE
GOTO511
188 IT=LT
IX(1,IR,IT)=-1
DO435I=1,IA
DO435J=1,JA
IP(I,J)=IX(I,J,IT)
IF(I.EQ.2)IP(I,J)=IXX(J,IT)
435 CONTINUE
IF(LAP.NE.1)GOTO50

C
C
C
'USED SPACE' KEYS INITIALIZED

LAP=0
LT=KAT
KAT=0
GOTO50
511 IF(LAP.NE.1)GOTO512
LAP=0
LT=KAT

```

```

      KAT=0
C
C      MIN C(J) DETERMINED
C
512 MCR=2147483646
      DO1L=1,LT
      DO1J=1,JAT
      IF(IX(1,J,L).EQ.0)GOTO1
      IF(MCR.GE.IXX( J,L))MCR=IXX( J,L)
      1 CONTINUE
C
C      SECTION CONTAINING MIN C(J) IDENTIFIED AND COLUMN
C      MARKED BY SETTING FIRST ROW VALUE TO ZERO
C
C      IT=SECTION IN WHICH MIN C(J) IS FOUND
C      IR=COLUMN CONTAINING MIN C(J)
C
      DO6I=1,LT
      IT=I
      DO6J=1,JAT
      IR=J
      IF(IX(1,J,I).EQ.0)GOTO6
      IF(MCR.EQ.IXX( J,I))GOTO7
      6 CONTINUE
      7 IX(1,IR,IT)=0
C
C      IF MIN C(J) IS GREATER THAN CURRENT VALUE OF C(J)
C      WHICH PRODUCED A FEASIBLE SOLUTION, THEN CURRENT
C      SOLUTION IS ALSO OPTIMAL AND PROBLEM IS SOLVED
C
      IF(IALT.EQ.1.AND.MCR.GE.ICJAY)GOTO888
      IF(IALT.EQ.1)GOTO557
      IF(IXX( IR,IT).GE.ICJAY)GOTO888
C
C      ALL SECTIONS CHECKED FOR FREE SPACE (I.E., ALL
C      COLUMNS MARKED)
C
557 IF(LT.EQ.1)GOTO69
      DO66I=2,LT
      DO67J=1,JAT
      IF(IX(1,J,I).NE.0)GOTO66
      67 CONTINUE
      GOTO68
      66 CONTINUE
      GOTO69
C
C      A TOTALLY MARKED SECTION HAS BEEN FOUND AND ITS
C      LOCATION WILL BE USED BY THE NEW SECTION TO BE
C      FORMED
C
      68 KAT=LT
      LT=I
      LKTT=LKTT+1
      LAP=1
      GOTO70
C
C      NO TOTALLY MARKED SECTION HAS BEEN FOUND AND THE TOTAL
C      SECTION COUNT IS INCREMENTED BY ONE
C
      69 LT=LT+1
      LKTT=LKTT+1
      70 IF(LKTT.GE.2000.AND.IOT.NE.1)GOTO171
      IF(LT.GE.451)GOTO171
C
C      NEW SECTION FORMED
C
      DO 450 I=1,IA
      DO 450 J=1,JA

```

```

      IP(I,J)=IX(I,J,IT)
      IF(I.EQ.2) IP(I,J)=IXX(J,IT)
450  CONTINUE
      IX(IA,IR,LT)=IP(IA,IR)+1
      DO1111 I=1,JA
1111  IX(1,I,LT)=MB(I)
C
C      C(J) FORMED , CHECKED FOR GREATER THAN CURRENT C(J)
C      WHICH PROVIDED A FEASIBLE SOLUTION AND MARKED
C      ACCORDINGLY
C
      DO91J=1,JAT
      IX(2,J,LT)=1
      IXX(J,LT)=IP(2,IR)+IXX(J,1)
C
C      IF C(J) IS GREATER THAN THE MAXIMUM COMPUTED COST
C      AFTER WHICH ALL COLUMNS OF FRACTIONAL PARTS WILL
C      BE DUPLICATES AND BOTH MODES OF COLUMN ELIMINATION
C      HAVE BEEN UTILIZED, PROGRAM STOPS.
C
      IF(IXX(J,LT).GE.ISUM .AND. IOT.NE.1) GOTO171
      IF(IXX(J,LT).GE.ISUM) GOTO172
C
      IF(IALT.NE.1 .AND. IXX(J,LT).EQ.ICJAY) IX(1,J,LT)=0
      IF(IXX(J,LT).GT.ICJAY .AND. LOKP.NE.1) IX(1,J,LT)=0
91  IF(IXX(J,LT).GT.NCJAY) IX(1,J,LT)=0
C
C      NON-BASIC VARIABLE VALUES CARRIED FORWARD
C
      DO10 I=1,JA
      IF(I.EQ.IR) GOTO10
      IX(IA,I,LT)=IP(IA,I)
10  CONTINUE
      DO 90 I=3,IAT
      DO 90 J=1,JAT
      IF(IX(1,J,LT).EQ.0) GOTO840
      IF(J.EQ.IR) GOTO85
C
C      X(J) GREATER THAN NB(J)? IF SO MARK ENTIRE COLUMN
C
      IF(IX(IA,J,LT).GE.NB(J)) GOTO81
      GOTO82
C
C      X(R) GREATER THAN NB(R)? IF SO MARK ENTIRE COLUMN
C
85  IXA=IX(IA,J,LT)+1
      IF(IXA.GE.NB(J)) GOTO81
      GOTO82
C
C      IF C(J) GREATER THAN CURRENT ICJAY, SET ENTIRE COLUMN
C      TO ZERO RATHER THAN COMPUTE
C
81  IX(I,J,LT)=0
      IX(1,J,LT)=0
      GOTO9C
C
C      REMAINING FRACTIONAL PARTS OF COLUMNS COMPUTED
C
82  ITP=IP(I,IR)+IX(I,J,1)
84  ALK=LK
      AIM=ITP
      BIM=AIM/ALK
      IF(BIM.LT.1) IX(I,J,LT)=ITP
      IF(BIM.EQ.1) IX(I,J,LT)=0
      IF(BIM.GT.1) IX(I,J,LT)=ITP-LK
      GOTO90
840  IX(I,J,LT)=0
90  CONTINUE
C
C      LAST C COLUMN (I.E., ALPHA VALUES) CARRIED FORWARD
C
      DC99 I=3,IA

```

```

99 IX(I,JA,LT)=IX(I,JA,1)
   IXX(JA,LT)=IXX(JA,1)

```

```

NEW SECTION COLUMNS CHECKED FOR DUPLICATES

```

```

IF(LAP.NE.1)KAT=LT
DO2J=1,JAT
DO2I=1,KAT
IF(I.EQ.LT)GOTO2
DO2K=1,JAT
IF(IXX(J,LT).NE.IXX(K,I))GOTO2
DO22L=3,IAT
RLK=IX(L,J,LT)-IX(L,K,I)
IF(ABS(RLK).LE.YLK)GOTO22
GOTO2

```

```

22 CONTINUE
   IF(IXX(J,LT).EQ.ICJAY.AND.IALT.EQ.1)GOTO205

```

```

DUPLICATE COLUMNS CHECKED TO DETERMINE WHETHER OR NOT
SUBSEQUENT SECTIONS FORMED WILL HAVE IDENTICAL
NON-BASIC VARIABLE VALUES IF THEY ARE RETAINED
(I.E., UNMARKED AND USED FOR NEW SECTIONS) ***

```

```

THIS STEP BYPASSED ON FIRST TRY FOR SOLUTION AND USED
ON SECOMD TRY IF ITERATIONS HAD EXCEEDED 2000
OR REQUIRED SECTION SPACE EXCEEDED 450.

```

```

IF(IOT.NE.1)GOTO218
205 DO 216 KD=1,JAT
   IF(KD.EQ.J.OR.KD.EQ.K)GOTO216
   IF(IX(IA,KD,I).NE.IX(IA,KD,LT))GOTO2
216 CONTINUE
   IXJLT=IX(IA,J,LT)+1
   IXKI=IX(IA,K,I)+1
   IF(J.EQ.K)GOTO217
   IF(IXJLT.NE.IX(IA,J,I))GOTO2
   IF(IXKI.NE.IX(IA,K,LT))GOTO2
   GOTO218
217 IF(IX(IA,J,LT).NE.IX(IA,K,I))GOTO2

```

```

ALL DUPLICATE COLUMNS MARKED ON FIRST TRY
ONLY DUPLICATES WHICH MEET REQUIREMENT '***' ABOVE
ARE MARKED ON SECOND TRY

```

```

218 IX(1,J,LT)=0

```

```

2 CONTINUE
   IF(LAP.EQ.0)KAT=0
   IF(LAP.EQ.1)GOTO71

```

```

IF USED SPACE NOT AVAILABLE DETERMINE WHETHER OR NOT
NEW SECTION COLUMNS TOTALLY MARKED...IF SO,REUSE
SPACE

```

```

DO911J=1,JAT
IF(IX(1,J,LT).NE.0)GOTO71
911 CONTINUE

```

```

   LT=LT-1
   GOTO511

```

```

50 IPP=1
   GOTO71

```

```

838 IOPP=1
   IOT=IOT+1
   IF(IOT.EQ.1.AND.IALT.EQ.1)GOTO881
   GOTO71

```

```

881 WRITE(6,882)

```

```

882 FORMAT('////' ALL EXISTING OPTIMAL SOLNS. SOLVED FOR ON
1ADDITIONAL OPTIMA NOW SEARCHED FOR USING FULL DUPLICAT

```

```

   GOTO711
171 IGT=IOT+1

```

```

      IOU=1
      CALLTIMEIT(-1,TIMEX)
      WRITE(6,542)TIMEX
542  FORMAT(///' TIME SPENT ON ATTEMPT AT INTEGER PORTION OF
      1,-6PF15.6,' SECONDS.')
```

IF(IOT.EQ.1.AND.IALT.NE.1)WRITE(6,543)

```

543  FORMAT(///' FIRST ATTEMPT FAILED, SOLN. MAY HAVE BEEN B
      1BLEM NOW RERUN USING TOTAL DUPLICATE CHECK.')
```

IF (IOT.EQ.1.AND.IALT.EQ.1.AND.MCR.LT.ICJAY.AND.
1IXX(J,LT).LT.ISUM)WRITE(6,3)

```

      IF(IOT.EQ.1.AND.IXX(J,LT).GE.ISUM)WRITE(6,1711)
1711 FORMAT(///' NO FEASIBLE SOLUTION AVAILABLE UNDER MODIFI
      1 COL. ELIMINATION METHOD..PROBLEM NOW RERUN USING TOTA
      1.')
```

3 FORMAT(///' ALTERNATE OPTIMA STILL BEING SEARCHED FOR W
1CE LIMITS REACHED..PROBLEM NOW RERUN USING TOTAL DUP.
IF(IOT.EQ.2.AND.IALT.EQ.1.AND.MCR.LT.ICJAY)WRITE(6,4)

```

      4 FORMAT(///' ALTERNATE OPTIMA STILL BEING SEARCHED FOR W
      1CE LIMITS REACHED.')
```

```

711  IF(IOT.EQ.1)WRITE(6,544)LKTT,LT
544  FORMAT(/' ITERATIONS THUS FAR= ',I5,', SECTIONS REQUIR
      GOTO71
172  JUMP=1
      ICT=IOT+1
71  RETURN
      END
```

```

SUBROUTINE FEAS(IR,M,N,IGNAT)

ROUTINE TO COMPUTE EACH POSSIBLE SOLUTION AND CHECK
FOR FEASIBILITY

COMMON/ONE/XUB(24),X(24),IB(14),AX(14),XLB(24),H(14),
1A(14,24),XI(14)
COMMON/THREE/KBVAR(10),KB(10),MX(13),MB(23),NB(10),
1AT(23,23),JX(13),AMX(13),MOX(22),IP(13,23),IXX(23,450)
1,IIP(22),NG(13),IX(13,23,450),IG(23,23)
INTEGER*2 IX,IG
COMMON/FOUR/LT,IA,JA,IT,NBV,LK,IPP,IAT,JAT,ICJAY,IOP,
1IBJAY,YLK,LKTT,LAP,KAT,IBV,LIP,IOU,MCR,ISUM,ICOST
1,JUMP,LOKP,LOKPT,NCJAY,MCJAY
DO1I=2,IAT
M=I-1
AMT=0.0
DO15J=1,JAT
IF(J.NE.IR)GOTO6
AP=IP(IA,J)+1
GOTO15
6 AP=IP(IA,J)
15 AMT=AT(M,J)*AP+AMT

BASIC VARIABLES VALUES

IF(I.NE.2)AMX(I)=AT(M,N)-AMT
IF(I.NE.2)GOTO1

OBJECTIVE FUNCTION VALUE

AMX(I)=AT(M,N)+AMT
1 CONTINUE
DO9I=2,IAT
IF(AMX(I).GE.0.0)GOTO2
B=AMX(I)-.5
GOTO3
2 B=AMX(I)+.5
3 JX(I)=B
IF(I.EQ.2)GOTO9
ARK=JX(I)
ALK=ABS(ARK)-ABS(AMX(I))
ABT=ABS(ALK)
IF(ABT.GE.0.0005)GOTO5
9 CONTINUE

BASIC VARIABLE BOUNDS CHECKED

DO4I=3,IAT
IQ=I-2
IF(JX(I).GT.KB(IQ))GOTO5
IF(JX(I).LT.0)GOTO5
4 CONTINUE
GOTO16
5 IOP=1
GOTO17
16 DO211I=1,IAT
211 MX(I)=JX(I)
LIP=LIP+1

FEASIBLE SOLUTION RECORDED

CALLTIMEIT(-1,TIMEX)
WRITE(6,542)TIMEX
542 FORMAT(//' COMPUTING TIME SINCE LAST POSSIBLE INTEGER
1-6PF15.6,' SECONDS.')
WRITE(6,400)LIP,LKTT
400 FORMAT(////1X14,' TH POSSIBLE FEASIBLE SOLUTION RECORD

```

```

        ITERATIONS REQUIRED THUS FAR...',I10)
        WRITE(6,3004)
3004  FORMAT(//' NON BASIC VARIABLES INTEGER SOLUTION VALUES
        DO212 I=1,JAT
        IF(I.EQ.IR)GOTO213
        MOX(I)=IX(IA,I,IT)
        WRITE(6,3002)MB(I),IX(IA,I,IT)
3002  FORMAT(/' X( ',I2,' )=',I6)
        GOTO212
    213  IBP=IX(IA,IR,IT)+1
        MOX(I)=IBP
        WRITE(6,3003)MB(I),IBP
3003  FORMAT(/' X( ',I2,' )=',I6)
    212  CONTINUE
        DO66 I=1,IBV
        LZ=I+2
    66  WRITE(6,300)KBVAR(I),MX(LZ)
    300  FORMAT(//' INTEGER SOLN. VALUE FOR BASIC VARIABLE, X('
    1)
        IF(IGNAT.EQ.1)MX(2)=MX(2)*(-1)
        WRITE(6,301) MX(2)
    301  FORMAT(///// ' OPTIMAL OBJECTIVE FUNCTION INTEGER SOLUTI
        CALLTIMEIT(0,TIMEX)
    17  RETURN
        END

```

```

SUBROUTINE TIMEIT(N,TIME)
C
C
C
N=0 STARTS CLOCK; N=-1 STOPS CLOCK
IT=N+2
GO TO (20,10),IT
10 CALL TIMON(M)
   TIME=M
   RETURN
20 CALL TIMOFF(M)
   TIME=M
   TIME=(TIME-M)*26.0
   RETURN
   END
C
C
C
ASSEMBLY LANGUAGE PROGRAM TO CONDUCT TIMING ROUTINE
TIMEALL CSECT
TIMON    ENTRY    TIMON,TIMOFF
        SAVE      (14,12)      ENTRY VIA -CALL TIMON(N)- 15.5
        USING     TIMON,12
        LR        12,15
        ST        13,TEMP1
        LA        13,SAVE1
        L          2,0(1,0)
        L          3,TOTIME
        ST        3,CLOCKR
        ST        3,0(2,0)
        STIMER    TASK,TUINTVL=CLOCKR
        L          13,TEMP1
EXIT     RETURN    (14,12),T,RC=0
TIMOFF   SAVE      (14,12)      ENTER VIA -CALL TIMOFF(N)-
        USING     TIMOFF,12
        LR        12,15
        ST        13,TEMP1
        LA        13,SAVE1
        L          2,0(1,0)
        TTIMER    CANCEL
        ST        0,0(2,0)
        L          13,TEMP1
        RETURN    (14,12),T,RC=0
        CNOP      0,4
TOTIME   DC        X'7FFFFFFF'
CLOCKR   DS        F
SAVE1    DS        18F
TEMP1    DS        F
        END

```

4

6

TIME TO COMPUTE CONTINUOUS SOLUTION = 0.019968 SECONDS.

RESULT	
X(1)	= 0.5082
X(2)	= 0.8197
X(3)	= 3.0492
X(4)	= 5.7377

HAVE $Z = -8.78688$ FRACTIONS

COMPUTING TIME SINCE LAST POSSIBLE INTEGER SOLUTION = 12.293632 SECONDS.

NON BASIC VARIABLES INTEGER SOLUTION VALUES...

X(5)= 7

X(6)= 4

X(7)= 1

X(8)= 0

X(9)= 0

INTEGER SOLN. VALUE FOR BASIC VARIABLE, X(2)= 0

INTEGER SOLN. VALUE FOR BASIC VARIABLE, X(4)= 0

INTEGER SOLN. VALUE FOR BASIC VARIABLE, X(3)= 0

INTEGER SOLN. VALUE FOR BASIC VARIABLE, X(1)= 0

OPTIMAL OBJECTIVE FUNCTION INTEGER SOLUTION=

7

ALL EXISTING OPTIMAL SOLUTIONS SOLVED FOR ON *COURSE* RUN..ADDITIONAL OPTIMA NOW SEARCHED
FOR USING *FINE* DUPLICATE COLUMN ELIMINATION.

ITERATIONS THUS FAR= 247, SECTIONS REQUIRED= 52

COMPUTATION TIME SINCE LAST POSSIBLE INTEGER SOLUTION = 3.840512 SECONDS.

ABOVE SOLUTION(S) REPRESENT UPPER/LOWER BOUNDS ON THE OPTIMAL SOLUTION..PROBLEM NOW
RERUN USING *FINE* DUPLICATE COLUMN ELIMINATION.

IF THIS METHOD FAILS OR IS INCOMPLETE DUE TO TIME/SPACE LIMITATIONS, THIS/THESE BOUNDS
IS/ARE THE BEST APPROXIMATION TO A SOLUTION WHICH THE PROGRAM CAN COMPUTE.

2.	3.	1.	2.	1.	0.	0.	3.
2.	2.	1.	2.	1.	0.	0.	0.
1.	0.	0.	0.	1.	0.	0.	-6.
18.	15.	0.	0.	1.	0.	0.	-7.
0.	0.	0.	0.	0.	0.	0.	0.
1.	0.	-1.	-1.	0.	0.	0.	0.
0.	0.	-1.	-1.	-1.	-1.	-1.	-1.
		0.	0.	0.	0.	0.	0.

TIME TO COMPUTE CONTINUOUS SOLUTION = 0.019968 SECONDS.

RESULT
 X(1) = 0.5082
 X(2) = 0.8197
 X(3) = 3.0492
 X(4) = 5.7377

Z = -8.78688
 HAVE FRACTIONS

TIME SPENT ON ATTEMPT AT INTEGER PORTION OF SOLUTION = 263.151616 SECONDS.

ALTERNATE OPTIMA STILL BEING SEARCHED FOR WHEN TIME/SPACE LIMITS REACHED.

	SECTION #1						
5	6	7	8	9	0		
11	51	46	40	35	-1608		
14	15	175	1	161	150		
98	105	127	7	29	135		
96	129	102	33	6	9		
16	174	17	158	1	93		
0	0	0	0	0	0		

COMMON DENOMINATOR USED TO COMPUTE FRACTIONAL PARTS = 183

SECTIONS REQUIRED EXCEEDS 450. TOTAL ITERATIONS THUS FAR = 768

HAVE FINISHED

BIBLIOGRAPHY

1. Greenberg, H., "A Dynamic Programming Solution to Integer Linear Programs," Journal of Mathematical Analysis and Applications, vol. 26, no. 2, May, 1969.
2. Dantzig, G. B., Linear Programming and Extensions. Princeton University Press, New Jersey, 1963.
3. Shapiro, J.F., "Dynamic Programming Algorithms for the Integer Programming Problem - I; the Integer Programming Problem Viewed as a Knapsack Type Problem", Operations Research, 16: 103-21, 1968.
4. Haldi, J., "25 Integer Programming Test Problems", Working Paper No. 43, Stanford University, Palo Alto, California, Dec. 1964, pp. 1-6, 12-13.
5. Gass, S. I., Linear Programming. McGraw-Hill Book Company, New York, 1964.
6. Gomory, R. E., "On the Relation Between Integer and Non-Integer Solutions to Linear Programs", Proceedings National Academy of Science, 53: 260-65, 1965.
7. Hillier, F. S. and Lieberman, G. J., Introduction to Operations Research. Holden-Day, Inc., San Francisco, California, 1968.
8. Muth, J. F. and Thompson, Gerald L., Industrial Scheduling. Prentice-Hall, Inc., New Jersey, 1963.
9. Hadley, G., Non-Linear and Dynamic Programming. Addison-Wesley Publishing Company, Inc., Reading, Massachusetts, 1964.
10. IBM, IBM System/360 Operating System, Supervisor and Data Management Services, IBM Corporation Programming Publications, San Jose, California, 1968.
11. IBM, IBM System/360 Operating System, Linkage Editor, IBM Corporation Programming Publications, San Jose, California, 1968.

INITIAL DISTRIBUTION LIST

	<u>No. Copies</u>
1. Defense Documentation Center Cameron Station Alexandria, Virginia 22314	20
2. Library, Code 0212 Naval Postgraduate School Monterey, California 93940	2
3. Commandant of the Marine Corps (Code A03C) Headquarters, U. S. Marine Corps Washington, D. C. 20380	1
4. James Carson Breckinridge Library Marine Corps Development and Educational Command Quantico, Virginia 22134	1
5. Professor Rex H. Shudde Department of Operations Analysis Naval Postgraduate School Monterey, California 93940	1
6. Professor Harold Greenberg Department of Operations Analysis Naval Postgraduate School Monterey, California 93940	1
7. Capt. J.C. Arick 14007 Cove Lane Apt. 202 Rockville, Maryland	1

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1 ORIGINATING ACTIVITY (Corporate author)

Naval Postgraduate School
Monterey, California 93940

2a. REPORT SECURITY CLASSIFICATION

Unclassified

2b. GROUP

3 REPORT TITLE

A Computer Program for Integer Solutions to Linear Programming Problems

4. DESCRIPTIVE NOTES (Type of report and, inclusive dates)

Master's Thesis; October 1969

5. AUTHOR(S) (First name, middle initial, last name)

John Chaney Arick

6. REPORT DATE

October 1969

7a. TOTAL NO. OF PAGES

72

7b. NO. OF REFS

11

8a. CONTRACT OR GRANT NO.

b. PROJECT NO.

c.

d.

9a. ORIGINATOR'S REPORT NUMBER(S)

9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)

10. DISTRIBUTION STATEMENT

This document has been approved for public release and sale; its distribution is unlimited.

11. SUPPLEMENTARY NOTES

12. SPONSORING MILITARY ACTIVITY

Naval Postgraduate School
Monterey, California 93940

13. ABSTRACT

An algorithm for the solution of integer linear programming problems is presented and programmed in Fortran IV for use on digital computers. The program incorporates an optional feature which provides all existing alternative optimal solutions. Solutions, computation times, and iteration requirements for each of thirteen test problems are summarized and discussed.

KEY WORDS

LINK A

WT

LINK B

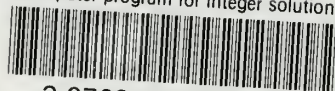
WT

LINK C

ROLE	WT
------	----

thesA66

A computer program for integer solutions



3 2768 001 00631 5

DUDLEY KNOX LIBRARY